University of Nottingham

Department of Manufacturing Engineering and Operations Management

MULTIPLE OBJECTIVE DECISION SUPPORT FRAMEWORK FOR CONFIGURING, LOADING AND RECONFIGURING MANUFACTURING CELLS

by

Adil BAYKASOĞLU, *B.Sc., M.Sc.*

Thesis Submitted to the University of Nottingham for the Degree of Doctor of Philosophy

April-1999

This thesis is dedicated to the memory of Atatürk and my fiancée, Emine.

ACKNOWLEDGEMENTS

There are a number of people and institutions that I would like to thank.

- *Professor Dr. Nabil N. Z. Gindy*, my supervisor, for his invaluable support, guidance and encouragement throughout the study.
- *Dr. Sameh M. Saad*, a good friend, for his invaluable concern and readiness to help.
- *Ms. Roxana Belecheanu,* a very good friend, for her help in checking the manuscript.
- *Carole, Badr, Sahana, Halit, Suleyman, Taylan, Steve, Nick, James, Alex, Thierry, Medhi* and many other very good friends, for their constant concern and friendship.
- *Professor Dr. Ali I. Sonmez* and *Dr. Turkay Dereli* for their moral support and encouragement.
- My Family and my fiancée, *Emine* for her great patience while I was away from her.
- The Turkish Higher Educational Council for their invaluable financial support.
- The University of Gaziantep for the grant.

TABLE OF CONTENTS

Acknowledgements

List of Figures vii

List of Tables x

Abbreviations xii

The list of papers produced from this research xiv

ABSTRACT A-1

CHAPTER ONE

1. INTRODUCTION 1-1

1.1 PROBLEM IDENTIFICATION AND SIGNIFICANCE 1-1

1.2 RESEARCH OBJECTIVES 1-5

1.3 THESIS ORGANISATION 1-6

CHAPTER TWO

2. REVIEW AND ANALYSIS OF THE LITERATURE 2-1

2.1 INTRODUCTION 2-1

2.2 OVERVIEW OF MODERN HEURISTIC OPTIMISATION TECHNIQUES 2-1

2.2.1 Genetic Algorithms (GA) 2-4

2.2.1.1 The General GA Algorithm 2-5

2.2.1.2 Some Example Applications of GA to Engineering Problems 2-5

2.2.2 Tabu Search (TS) 2-8

2.2.2.1 The General TS Algorithm 2-9

2.2.2.2 Some Example Applications of TS to Engineering Problems 2-1

2.2.3 Simulated Annealing (SA) 2-1

2.2.3.1 The General SA Algorithm 2-1

2.2.3.2 Some Example Applications of SA to Engineering Problems 2-1

2.3 MULTIPLE OBJECTIVE DECISION MAKING 2-1

2.3.1 Pareto Optimality 2-1

2.3.2 Techniques for Multiple Objective Optimisation 2-19
2.3.2.1 Utility Function Formulation 2-20
2.3.2.2 Global Criterion Formulation 2-22
2.3.2.3 Goal Attainment Method 2-22
2.3.2.4 Bounded Objective Function Formulation 2-24
2.3.2.5 Game Theoretical Method 2-24
2.3.2.6 Lexicographic (constraint) Method 2-26
2.3.2.7 Genetic Algorithms 2-27
2.3.2.7.1 Population based non-Pareto approach 2-27
2.3.2.7.2 Pareto based approach 2-28
2.3.2.7.3 MOGA: Multi objective genetic algorithm 2-28
2.3.2.8 Goal Programming Method 2-30
2.4 MANUFACTURING SIMULATION AND SIMULATION OPTIMISATION 2-32
2.4.1 Methodology of a Simulation Study: A Short Review 2-33
2.4.2 Simulation with SIMAN 2-36
2.4.3 Simulation and Optimisation 2-39
2.5 CELLULAR MANUFACTURING AND CELL FORMATION 2-44
2.6 LOADING AND SCHEDULING IN CELLULAR MANUFACTURING 2-50
2.6.1 Cell Loading 2-51
2.6.2 Cell Scheduling 2-53
2.7 RECONFIGURATION ISSUES IN CELLULAR MANUFACTURING 2-55
2.7.1 Problems Associated with Cellular Manufacturing Systems 2-55
2.7.2 The Reconfiguration concept 2-56
2.7.3 Reconfiguration from the Cellular Manufacturing Viewpoint 2-58
2.9 CONCLUSIONS OF THE LITERATURE REVIEW 2-60

CHAPTER THREE

3. AN OVERVIEW OF THE PROPOSED FRAMEWORK 3-1
3.1 INTRODUCTION 3-1
3.2 AN OVERVIEW OF THE RECONFIGURATION PROBLEM IN CMS 3-2

3.3 THE PROPOSED FRAMEWORK 3-4
3.3.1 Production Planning 3-7
3.3.2 Generic Process Planning 3-8
3.3.3 Cell Configuration 3-8
3.3.4 Cell Loading 3-8
3.3.5 Parametric Simulation System 3-9
3.3.6 Multiple Objective Optimisation 3-10
3.3.7 Reconfiguration (Virtual Cell Formation) 3-10
3.4 CONCLUSIONS 3-11

CHAPTER FOUR

4. APPLICATION OF TABU SEARCH TO THE GENERAL PROBLEM OF MULTIPLE OBJECTIVE OPTIMISATION: DETERMINATION OF PARETO-OPTIMAL SET 4-1
4.1 INTRODUCTION 4-1
4.2 TABU SEARCH BASED APPROACH TO FIND PARETO OPTIMAL SET IN MOO 4-2
4.2.1 Development of a Tabu Search Algorithm 4-3
4.2.2 Numerical Examples and Comparative Work 4-15
4.3 CONCLUSIONS 4-23

CHAPTER FIVE

5. DEVELOPMENT OF TABU SEARCH ALGORITHM TO SOLVE PRE-EMPTIVE GOAL PROGRAMMING MODELS 5-1
5.1 INTRODUCTION 5-1
5.2 A TABU SEARCH BASED APPROACH TO SOLVE PRE-EMPTIVE GOAL PROGRAMMING MODELS 5-2
5.2.1 Preemptive Goal Programming 5-3
5.2.2 The TS Algorithm 5-5
5.2.3 Numerical Examples and Comparative Work 5-9
5.2.4 Application of the proposed algorithm to simulation optimisation 5-1
5.3 CONCLUSIONS 5-1

CHAPTER SIX

6. DEVELOPMENT OF MULTIPLE OBJECTIVE MANUFACTURING CELL FORMATION MODULE 6-1

6.1 INTRODUCTION 6-1

6.2 MATHEMATICAL FORMULATION OF THE MULTIPLE OBJECTIVE CELL FORMATION PROBLEM 6-3

6.3 APPLICATION OF MULTIPLE OBJECTIVE TABU SEARCH ALGORITHM TO SOLVE THE MATHEMATICAL MODEL 6-15

6.4 EXAMPLE APPLICATIONS AND THE COMPARATIVE WORK 6-17

6.5 CONCLUSIONS 6-26

CHAPTER SEVEN

7. DEVELOPMENT OF MULTIPLE OBJECTIVE CELL LOADING MODULE 7-1

7.1 INTRODUCTION 7-1

7.2 PROBLEM STATEMENT 7-3

7.3 MODELLING AND SOLUTION APPROACH 7-8

7.3.1 Mathematical Modelling 7-10

7.3.2 Application of Multiple Objective Tabu Search Algorithm to Solve Loading Model 7-12

7.3.3 Simulation and Scheduling Module (SSM) 7-14

7.3.4 Generation of the Loading and Scheduling Scenario (The simulation optimisation cycle) 7-17

7.4 EXPERIMENTAL WORK 7-18

7.5 CONCLUSIONS 7-27

CHAPTER EIGHT

8. DEVELOPMENT OF MULTIPLE OBJECTIVE RECONFIGURATION MODULE 8-1

8.1 INTRODUCTION 8-1

8.2 THE RECONFIGURATION STRATEGY 8-4

8.3 MATHEMATICAL MODELLING OF VIRTUAL CELL FORMATION PROBLEM 8-5
8.3.1 Application of Multiple Objective TS Algorithm to Solve Virtual Cell Model 8-8
8.4 EXPERIMENTAL WORK 8-11
8.5 CONCLUSIONS 8-18

CHAPTER NINE

9. SUMMARY, CONCLUSIONS AND FURTHER RESEARCH 9-1
9.1 INTRODUCTION 9-1
9.2 SUMMARY OF THE THESIS 9-1
9.3 THE RESEARCH CONTRIBUTIONS 9-3
9.4 SUGGESTIONS FOR FUTURE WORK 9-13

REFERENCES AND BIBLIOGRAPHY R-1

APPENDICES

Appendix I : C/C++ code for Example 2 of Chapter 5 I-1

Appendix II : Additional test problems for Chapter 5 II-5

Appendix III : Resource Elements III-11

Appendix IV : FORTRAN-90 code for the cell configuration module IV-21

Appendix V : Application of MOCACEF 1.0 V-49

Appendix VI : Frames for loading and reconfiguration modules VI-58

Appendix VII : Continuation of the experimental work from Chapter 9 VII-61

LIST OF FIGURES

Chapter-1

Figure 1.1 Research areas and their inter-relationships 1-8

Chapter-2

Figure 2.1 A pictorial comparison of classical and heuristic optimisation strategies 2-3
Figure 2.2 Graphical explanation of Pareto optimality 2-17
Figure 2.3 The search directions in VEGA 2-28
Figure 2.4 Various search directions of MOGA 2-30
Figure 2.5 The simulation cycle 2-34
Figure 2.6 SIMAN software organisation 2-38
Figure 2.7 Three different approaches to relate part processing requirements to the available machine tools 2-49

Chapter-3

Figure 3.1 Changes in cellular manufacturing system design in relation the changing production requirements 3-3
Figure 3.2 The proposed integrated framework for reconfiguring CMSs 3-5
Figure 3.3 Parametric simulation optimisation system 3-9
Figure 3.4 Formation of VCs in relation to changing production requirements 3-11

Chapter-4

Figure 4.1 The flowchart of the basic TS algorithm 4-3
Figure 4.2 Neighbourhood move spaces of different move approaches 4-6
Figure 4.3 The flowchart of the TS algorithm to find Pareto optimal solutions 4-11
Figure 4.4 Part of a step by step manual solution for the example problem 4-12
Figure 4.5 Computer simulation results for the test problem 4-13
Figure 4.6 Behaviour of the algorithm while searching for Pareto optimal set in MOO 4-14

Figure 4.7 Computer simulation result for Example 1 4-18
Figure 4.8 Computer simulation result for Example 2 4-20
Figure 4.9 Graphical comparison of extreme points obtained from PSM, GA and MOTS 4-22
Figure 4.10 Computer simulation result for Example 3 4-23

Chapter-5

Figure 5.1 Step by Step manual solution of the example problem 5-9
Figure 5.2 The flowchart of simulation optimisation strategy 5-16

Chapter-6

Figure 6.1 Analysis of capacity deviations in a cell 6-10
Figure 6.2 Neighbour solutions generation 6-16
Figure 6.3 Comparison of extra resource requirement to configure cells 6-24
Figure 6.4 Comparison of cell utilisation levels 6-25

Chapter-7

Figure 7.1 Components of loading problem in CMS 7-2
Figure 7.2 Graphical description of the CMS loading problem 7-4
Figure 7.3 An example string for a part type and determination of its alternative cells assignment (machining time data is not shown in the string) 7-5
Figure 7.4 Construction of the MIG graph for a hypothetical CMS with eight part types and three manufacturing cells 7-6
Figure 7.5 A simplified flowchart of the loading system 7-9
Figure 7.6 Neighbour solutions generation 7-13
Figure 7.7 Simulation and scheduling module 7-17
Figure 7.8 Parametric simulation system 7-18
Figure 7.9 Layout of the prototype CM system 7-19
Figure 7.10 The conversion behaviour of the performance measures considered in the case study 7-25
Figure 7.11 Part assignment scenario generated by the loading system 7-26

Figure 7.12 A part of production schedule generated by the loading system 7-27

Chapter-8

Figure 8.1 The conversion behaviour of the performance measures 8-15
Figure 8.2 Output of reconfiguration module as the new set of virtual cells and corresponding part assignment 8-16
Figure 8.3 A small part of production schedule generated automatically for the new set of virtual cells 8-17
Figure 8.4 Performance improvements after reconfiguration 8-17

Appendix III

Figure III.1 FGSs in a classical lathe III-13
Figure III.2 FGSs in a milling machine III-14
Figure III.3 RE based representation of a machining facility with three machines III-15
Figure III.4 REs based representation of the manufacturing cell III-20

Appendix VII

Figure VII.1 Conversion behaviour of the loading module VII-63
Figure VII.2 Part assignment scenario for the best solution found VII-64
Figure VII.3 Conversion behaviour of the reconfiguration module VII-66
Figure VII.4 The output of the reconfiguration module: new VCs and corresponding part assignments VII-67
Figure VII.5 A portion of the production schedule generated automatically by the reconfiguration module VII-68

LIST OF TABLES

Chapter-3

Table 3.1 An example assumed part list 3-7

Chapter-4

Table 4.1 Comparison of extreme points obtained from plain stochastic method, genetic algorithms and multiple objective TS 4-22

Chapter-5

Table 5.1 Solution summary for Example 1 5-11
Table 5.2 Solution summary for Example 2 5-12
Table 5.3 Solution summary for Example 3 5-14

Chapter-6

Table 6.1 Machine tools and their capabilities based on REs in the test model 6-18
Table 6.2 Part data: processing time (min), total demand, processing sequences 6-18
Table 6.3 The output summary of MOCACEF 1.0 6-19
Table 6.4 Part-machine matrix 6-20
Table 6.5 The initial solution obtained from the *p-median* model 6-21
Table 6.6 The final solution for the *p-median* model after eliminating inter-cell movements 6-21
Table 6.7 Solution summary for p-median model 6-22
Table 6.8 The initial solution obtained from Seifoddini and Wolfe (1986)'s method 6-23
Table 6.9 Final solution for Seifoddini and Wolfe (1986)'s method 6-23
Table 6.10 Solution summary for Seifoddini and Wolfe (1986)'s method 6-24

Chapter-7

Table 7.1 Dispatching rules 7-15
Table 7.2 Due date assignment rules 7-16
Table 7.3 Virtual cell capabilities based on *REs* 7-20

Table 7.4 Part list: generic part process plans 7-20

Appendix III

Table III.1 The capability matrix M_k for the example manufacturing cell III-19

Table III.2 Clustering FGSs to form REs for the example manufacturing cell III-19

Appendix V

Table V.1 Machines and their capabilities based on REs V-49

Table V.2 Annual machine capacity V-50

Table V.3 Parts processing requirements as generic process plans based on REs V-51

Table V.4 RE based part processing sequence data V-52

Table V.5 Part processing time and annual demand V-53

Table V.6 Output of the MOCACEF 1.0 V-54

Appendix VII

Table VII.1 The new part list and the correspond generic process plans VII-61

ABBREVIATIONS

AGV	Automated Guided Vehicle
BMS	Biological Manufacturing Systems
CIM	Computer Integrated Manufacturing
CM	Cellular Manufacturing
CMS	Cellular Manufacturing Systems
CNC	Computer Numerical Control
exp	Exponential Function (e)
FGS	Form Generating Schema
FMS	Flexible Manufacturing Systems
GA	Genetic Algorithms
GP	Goal Programming
GT	Group Technology
HMS	Holonic Manufacturing Systems
JIT	Just in Time
M/C	Machine Tool
max	Maximum
min	Minimum
MIG	Minimum Interaction Graph
MOO	Multiple Objective Optimisation
MOTS	Multiple Objective Tabu search
MOCACEF 1.0	Multiple Objective Capability based Cell Formation Tool Version 1
MRP	Material Requirements Planning
OMA	Optimisation Modelling Approach
PGP	Pre-emptive Goal Programming
PSM	Plain Stochastic Method
RE	Resource Elements
RMS	Response Surface Methodology
SA	Simulated Annealing
SSM	Simulation and Scheduling Module

TS	Tabu Search
VC	Virtual Cells
VCMS	Virtual Cellular Manufacturing Systems

THE LIST OF PAPERS PRODUCED FROM THIS RESEARCH

1. Baykasoglu, A., Owen, S. and Gindy, N., (1999) Solution of goal programming models using a basic taboo search algorithm. Accepted for publication in *Journal of Operational Research Society*. (It will be published in the September-1999 issue)

2. Baykasoglu, A., Owen, S. and Gindy, N., (1999) A taboo search based approach to find the pareto optimal set in multiple objective optimisation. Accepted for publication in *J. of Engineering Optimization*. (It will be published in the August-1999 issue 'vol:31-no:6')

3. Baykasoglu, A., Gindy, N., (1999) MOCACEF 1.0: Multiple objective capability based approach to form part-machine groups for cellular manufacturing applications. Accepted for publication in *Int. J. of Production Research*.

4. Saad, S. M., Baykasoglu, A., Gindy, N., (1999) A new integrated system for loading and scheduling in cellular manufacturing. Submitted to *Production Planning and Control*.

5. Baykasoglu, A., Saad, S. M., Gindy, N., (1998) A loading approach for cellular manufacturing systems, *FAIM'1998: 8th International Conference on Flexible Automation and Intelligent Manufacturing*, July 1-3 1998, Portland, Oregon, USA, pp. 215-226.

6. Baykasoglu, A., Gindy, N., Saad, S. M., (1998) A framework for the reconfiguration of cellular manufacturing systems. *IMS-98, 2nd Int. Symposium on Intelligent Manufacturing Systems*, 6-7 August 1998, Sakarya, Turkey, pp. 565-573.

7. Baykasoglu, A., Gindy, N., (1999) Loading flexible cells: Tabu search based simulation optimisation approach, paper to be appeared in the *15th International Conference on Production Research*, 9-13, August-1999, Limerick, Ireland.

8. Baykasoglu, A., Saad, S. M., Gindy, N., (1999) A study on loading cellular manufacturing systems with multiple objectives. To be submitted to *Int. J. of Computer Integrated Manufacturing*

9. Gindy, N., Baykasoglu, A. and Saad, S. M., (1999) An Integrated Framework for Reconfiguration of Cellular Manufacturing Systems Using Virtual Cells. To be submitted to *Production Planning and Control.*

ABSTRACT

The potential advantages of Cellular Manufacturing Systems (CMS) are very well known in industry. However it is also shown that their performance is very sensitive to changing production requirements. The detrimental effects of changing production requirements on the performance of CMS can be alleviated by "implementing better manufacturing cell designs", "employing effective part loading strategies" and "reconfiguration".

This thesis proposes a decision support framework that provides solution strategies for manufacturing cell design, cell loading and reconfiguration problems. There are three main modules in the proposed framework, named as *cell formation*, *loading* and *reconfiguration*. Each module can handle multiple objectives and integrates several planning and design functions, by considering the capabilities of manufacturing resources. Reconfiguration decisions are made explicitly in the proposed framework by answering the questions "*when to reconfigure?*" and "*how to reconfigure?*". In order to answer these questions, the modules of the proposed framework are interconnected. The cell formation module creates the initial set of cells. The loading module makes the 'part to cell assignment' and the scheduling in each production period. The reconfiguration module regenerates manufacturing cells, if the loading module can not find a satisfactory solution.

The cell formation module solves the part-machine cell formation problem by simultaneously considering multiple objectives and constraints. Overlapping machine capabilities and generic part process plans are taken into account in the model formulation. A new approach for the evaluation of machine capacities is also presented. Results of the comparative study show that the proposed cell formation method gives better results than several other cell-formation procedures. The manufacturing cells are formed with improved capacity utilisation levels and reduced

extra machine requirements. The method is also more likely to produce independent manufacturing cells with higher flexibility.

The loading module solves the 'part to cell assignment' and 'cell scheduling' problems simultaneously for cellular manufacturing applications. Alternative parts to cell and machine assignments are considered by making use of generic part process plans in the model formulation. A parametric simulation model is developed to determine cell schedules for a given part assignment scenario. The proposed loading system can assess performance of the CMS in each production period. Therefore a decision can be made about its reconfiguration. It is also shown that the efficiency of CMSs facing changing production requirements can be improved and/or sustained by using the proposed loading strategy.

The reconfiguration module takes the existing cell configuration as the current solution and generates a new solution from it, to enhance its performance. The model is objective driven and considers multiple objectives and constraints within a goal programming framework. The *virtual cell* concept is applied as the reconfiguration strategy. In the *virtual cell* approach the physical locations of machines are not changed, only cell memberships of machines are updated after reconfiguration. The results of the test studies showed that it is possible to improve the performance of CMS by reconfiguring it using virtual cells.

The cell formation, loading and reconfiguration problems issues discussed in this thesis are combinatorially complex multiple objective optimisation problems. Additionally simulation is used to evaluate several of the objective functions used in the modelling of loading and reconfiguration problems. Classical optimisation algorithms have various limitations in solving such problems. Therefore, Tabu Search (TS) based multiple objective optimisation algorithms are developed. The proposed TS algorithms are general-purpose and can also be used to solve other multiple objective optimisation problems. The results obtained from several test problems show the proposed TS algorithms to be very effective in solving multiple objective optimisation problems. More than 50% improvement in solution quality is obtained in some test problems.

CHAPTER ONE

1. INTRODUCTION

1.1 PROBLEM IDENTIFICATION AND SIGNIFICANCE

Today's manufacturing environment is mainly characterised by rapid and frequent changes. In such an environment, production is generally made to order, the preferences of customers are hard to forecast and product mix changes regularly and in some cases drastically. In these circumstances, manufacturing companies must adapt to the changing requirements of the market forcing them to increase flexibility and responsiveness.

Cellular Manufacturing Systems (CMS) have been accepted as modern solutions to many of today's market requirements. During the last two decades, their popularity has increased. Industrial applications have also proven that it is virtually impossible to implement a large-scale automated manufacturing system without using cellular concepts (Kusiak, 1990).

CMS have many theoretically and practically proven advantages (Kusiak and Chow, 1988, Burbidge, 1987, Wemmerlov and Johnson, 1997). These include:

- Reduction in production lead time
- Reduction in work-in-process
- Reduction in labour

- Reduction in set up time and tooling
- Reduction in rework and scrap materials
- Superior quality and productivity
- Reduction in order time delivery
- Reduction in control effort
- Reduction in scheduling and production planning complexity
- Improvement in human relations
- Reduction in paper work

Additionally, cellular manufacturing is one of the best approaches for implementing advanced manufacturing technologies like CIM, FMS and JIT (Gallagher and Knight, 1986, Reisman *et. al.*, 1997).

However, CMSs also have some difficulties or disadvantages. The majority of industrial applications have shown that most of the benefits are obtained from organisations that have a degree of part standardisation and moderate batch sizes (Marsh, *et. al.*, 1997). The main problem areas frequently mentioned in the literature include:

- CMS performance is sensitive to changes in product design, part mix, demand etc. (Sasani, 1990, Seifoddini and Djassemi, 1996,1997).
- To ensure that parts are completely processed within a cell, to eliminate inter-cell movement and related control and quality problems, some machines may have to be duplicated i.e. additional capital investment is necessary (Irani *et. al.*, 1993).
- Since there is an overall increase in the number of machines, as compared with a job shop, typically the average machine utilisation is lower. In addition, since production volumes for part families rarely correspond to integer machine

requirements, duplication of the same machines among different cells could result in poor capacity utilisation and higher production costs. When the part-mix changes, an imbalance in cell loading happens which leads to an imbalance in machine utilisation. In other words, since the shop cannot even respond to short-time fluctuations in demand, dedicating machines to specific parts typically results in unbalanced utilisation (Vakharia and Kaku, 1994, Dahel and Smith 1993).

- Cell systems are less flexible than functional job-shops, due to division of the shop into a number of independent cells. As a result, their performance can actually worsen unless properly designed (Vakharia and Kaku, 1994, Abedzadeh and O'Brien, 1996).
- Due to the costs of machine relocation, cell systems are costly to construct (Adil et. al., 1996).

Based on the investigations in the literature (Chapter 2), it is noticed that some of the important reasons behind the performance problems of CMSs facing changing production requirements are generally the outcome of the following (Wemmerlov and Hyer, 1987, 1989, Wemmerlov and Johnson, 1997):

- *Improper initial design of manufacturing cells*: Most cell formation techniques form part and machine groups based on only one criterion. The criterion is generally defined as maximising similarity or minimising inter-cell movement without considering many other parameters, like processing sequences, machine capacities, load balance etc. These parameters may have important effects on the performance of generated cell designs (Wemmerlov and Johnson, 1997). Therefore, more advanced cell formation procedures are required for generating better cellular manufacturing configurations.

- *Unavailability of effective cell loading strategies*: In cellular manufacturing applications, it is generally assumed that for any part type there is only one feasible cell that is initially designed for the family of this part type, and one process plan is available (Choobineh, 1984, Liang and Dutta, 1990). However, in a dynamic manufacturing environment part spectrums change over time and the performance of a CMS is closely related to the effective assignment of parts to its cells and their scheduling that is known as *loading* (Greene and Sadowski, 1986, Baykasoglu *et. al.*, 1998-a). The loading problems of CMSs have not received serious attention in the literature. In order to effectively operate CMSs facing changing production requirements, *loading problems* should be addressed and studied further.
- *Unavailability of a reconfiguration strategy*: Even if initial manufacturing cells are designed successfully and effective loading strategies are employed, in some production periods, occurrences of low performance may not be prevented. In such cases existing manufacturing cells should be reconfigured to improve the performance of CMSs.

In order to improve the performance and retain the advantages of CMSs in a dynamic manufacturing environment, the issues mentioned above should be explored. The literature review of this research (Chapter 2) has also indicated that, there is a need to employ and/or develop heuristic optimisation procedures for solving cell formation, loading and reconfiguration problems due to limitations of classical optimisation procedures (Dhingra and Lee, 1994). The importance of a framework which can integrate various production functions, like process planning and manufacturing

system simulation has also been suggested by researchers to improve performance of manufacturing systems (Liles and Huff, 1990, Song and Hitomi, 1996).

1.2 RESEARCH OBJECTIVES

The main objectives of this research are to develop a cell formation technique and a reconfiguration framework for cellular manufacturing applications.

The main purpose of developing a cell formation technique is to generate a good initial shop floor configuration by considering multiple design objectives and constraints. As discussed in Chapter 2, the majority of existing cell formation procedures are based on single design criteria and they do not consider many important parameters, like alternative machines for part processing, part processing sequences, machine capacities etc., which are necessary for an effective cell design.

The main target in developing a reconfiguration technique is to ensure that the CMS is working under the "best possible conditions". Reconfiguration of a CMS is a complex problem that has received little attention in the available literature. In order to make a decision about reconfiguration, the performance of the CMS should be determined. This requires solution of the cellular manufacturing loading problem, which has also received little attention in the literature. It is observed that cell formation, loading and reconfiguration problems are interconnected. Therefore one of the aims of this work is to develop a framework that can be used as a decision support tool for solving these problems. Cell formation, loading and reconfiguration problems can be better solved if several production functions like process planning and manufacturing simulation are integrated. One of the objectives of this thesis is to

integrate some of these functions to achieve a robust cell formation, loading and reconfiguration framework.

Many of the problems mentioned above have some of the following characteristics:

- They have multiple objectives.
- They have multiple constraints.
- They are known complex combinatorial optimisation problems.

The applicability of classical optimisation procedures is limited for these types of problems (Dhigra and Lee, 1994). Therefore, the research also aims to develop effective heuristic optimisation procedures which can be used for solving multi-objective, multi-constraint complex cell formation, loading and reconfiguration problems in cellular manufacturing applications.

1.3 THESIS ORGANISATION

The remaining chapters of this thesis are organised as follows:

A detailed review and analysis of the literature related to the topics being studied in this thesis is presented in Chapter 2.

An overview of the proposed integrated multiple objective decision-support framework for configuring, loading and reconfiguring CMSs is given in Chapter 3.

The development of Tabu search (TS) based general purpose multiple objective optimisation (MOO) algorithms, and their application to some test problems. are presented in Chapters 4 and 5. Chapter 4 presents the solution of the general problem

of MOO i.e. Pareto Optimality by the proposed TS algorithm. Chapter 5 explains the solution of pre-emptive goal programming models by the proposed TS algorithm. Pre-emptive goal programming is used for modelling multiple objective cell formation, loading and reconfiguration problems.

The development of a multiple objective capability based manufacturing cell formation model is presented in Chapter 6.

The proposed integrated model for loading cellular manufacturing system with multiple objectives is presented in Chapter 7. This chapter also explains the development of the TS based parametric simulation optimisation algorithm, which is used in the loading and reconfiguration of CMSs.

The proposed strategy and algorithm for the reconfiguration of cellular manufacturing systems is explained in Chapter 8.

Conclusions are given in Chapter 9, together with suggestions for future research. Supporting material is included in the Appendices.

The Figure 1.1 below shows various work areas, which have been researched in this project and their inter-relationships.

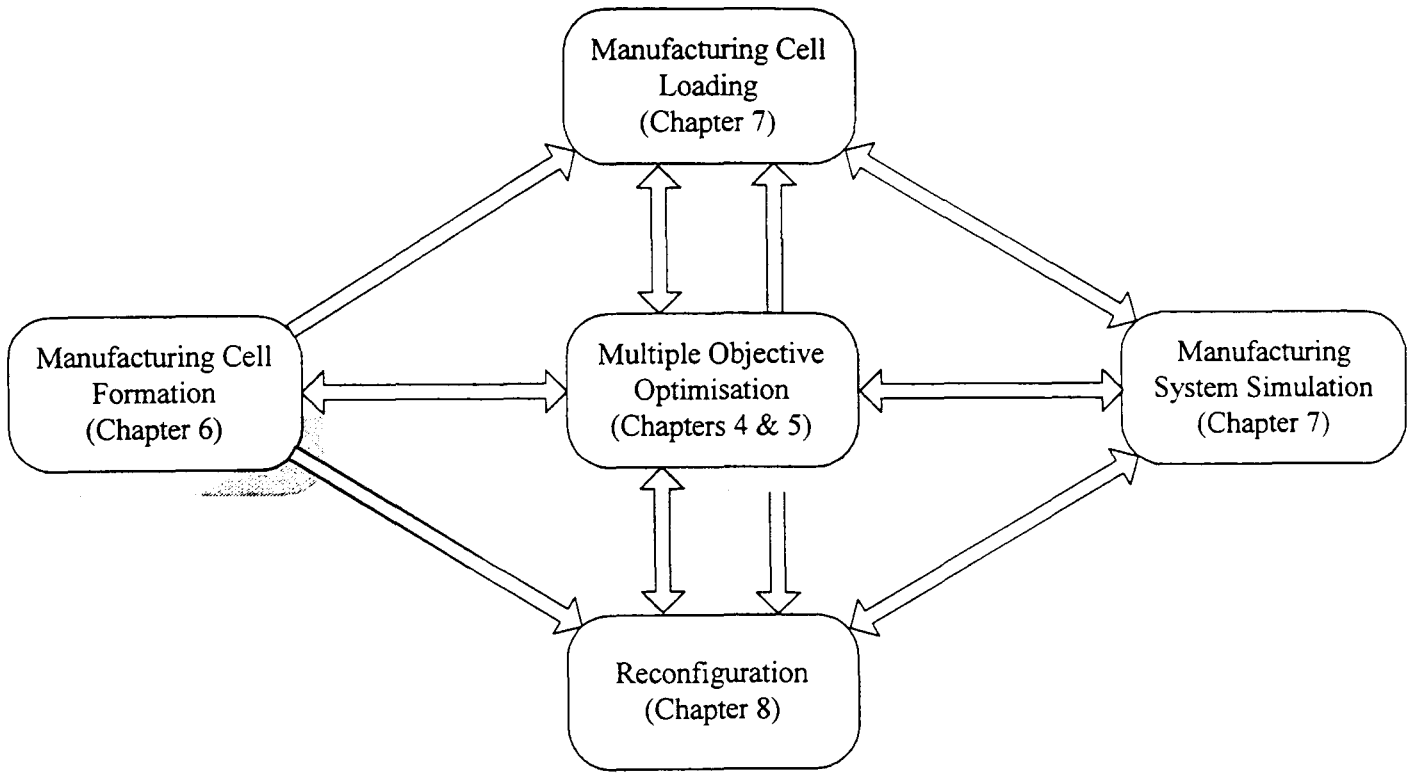


Figure 1.1 Research areas and their inter-relationships

CHAPTER TWO

2. REVIEW AND ANALYSIS OF THE LITERATURE

2.1 INTRODUCTION

This chapter presents a detailed survey of the relevant literature related to the study reported in this thesis. The content of the thesis relates to a number of different topics, namely, heuristic techniques in optimisation and multiple objective optimisation, cellular manufacturing, cell design, reconfiguration, loading and manufacturing system simulation. The thesis contributes to these domains, therefore each section in this chapter is devoted to one or more of these areas.

2.2 OVERVIEW OF MODERN HEURISTIC OPTIMISATION TECHNIQUES

Many engineering problems (design, planning, control etc.) can be cast as optimisation problems (Fogel, 1995). The fundamental process of optimisation begins with some candidate solutions and iteratively refines and improves them through various procedures.

Classical optimisation methods limit the usage of *Optimisation Modelling Approach* (OMA) to model and solve many real life engineering problems (Dhingra and Lee, 1994). This is mainly due to their inherent solution mechanisms. Their solution procedures are built upon the type of objectives, constraint functions (linear, non-linear, posynomial, signomial etc.) and variables (integer, real, binary etc.) used in

the problem formulation (Schoenauer and Xanthakis, 1993, Smith and Tate, 1993, Michalewicz, *et. al.*, 1996, Yokota, *et. al.*, 1996). Their efficiency is also strictly dependent on the structure of the solution space (convex, non-convex, uniform, non-uniform, continuous, discontinuous etc.), size of the solution space and number of variables and constraints used in problem formulation (Homafair, *et. al.*, 1994, Fogel, 1995, Lin and Hajela, 1992, Wienholt, 1993). They also do not offer a general solution strategy that can be applied to a large number of problem formulations in which different type of variables, objectives and constraint functions are used (Malasri, *et. al.*, 1996). For example, the *Simplex Algorithm* can be used to solve models with linear objective and constraint functions, *Branch and Bound Algorithms* can be used to solve linear models with integer variables. *Geometric programming* can be used to solve non-linear models with a posynomial or signomial objective function etc.

Many engineering problems require usage of different types of variables, objective and constraint functions simultaneously in their formulation. Therefore, classical optimisation procedures are generally not adequate for their solution (Man, *et. al.*, 1996, Zhang and Wang, 1993, Lin and Hajela, 1992). Researchers have exported great effort to adapt many engineering problems to these classic optimisation procedures. Many diverse example applications can easily be found in the literature. Some specific applications from the author's previous research relating to scheduling, process planning and cutting conditions optimisation can be referred to as examples (Baykasoglu, 1995, Kayacan, *et. al.*, 1996, Sonmez, *et. al.*, 1999, Sonmez and Baykasoglu, 1998, Sonmez, *et. al.*, 1996, Filiz, *et. al.*, 1996). It is very hard to formulate a real life problem that suits a specific solution procedure. In order to

achieve this, it is necessary to make some modifications and/or simplifying assumptions to the original problem parameters (rounding variables, softening constraints etc.) during formulation. This affects the solution quality (Michalewicz, *et. al.*, 1996).

A new set of efficient, problem and model independent general purpose heuristic optimisation techniques were proposed by researchers to overcome drawbacks of classical optimisation procedures and to enable efficient use of OMA. These techniques should be flexible and modifiable and/or adaptable to suit specific problem requirements (Chipperfield, *et. al.*, 1994) (see Figure 2.1). Three of these widely accepted and applied techniques, namely: *Genetic Algorithms*, *Tabu Search* and *Simulated Annealing* are briefly explained in the following sub-sections. These techniques are still maturing and are attracting wide research interest.

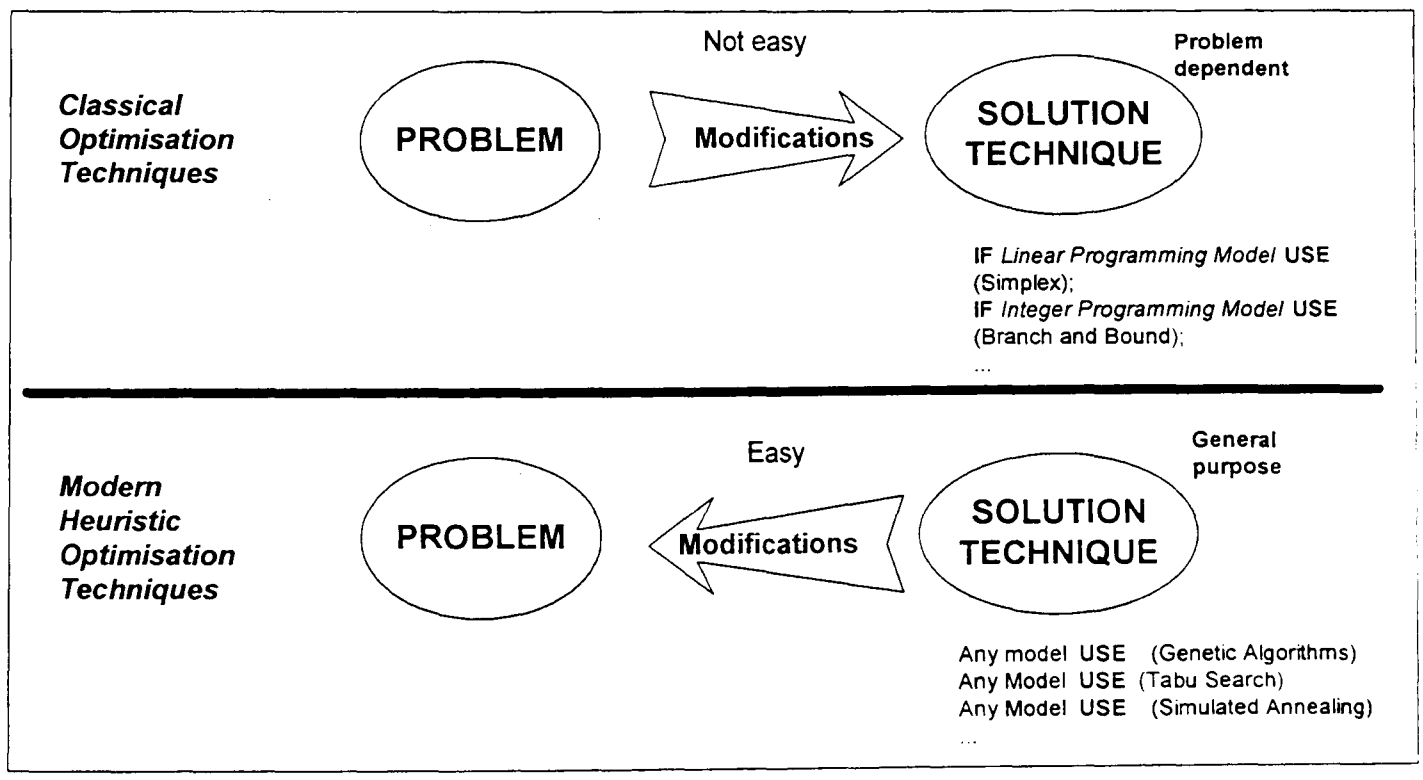


Figure 2. 1 A pictorial comparison of classical and heuristic optimisation strategies

2.2.1 Genetic Algorithms (GA)

Genetic Algorithms are adaptive search methods based on an abstract model of natural evolution. GAs were developed by Holland in the 1970s and only recently has their potential for solving optimisation problems been explored (Michalewicz, 1996, Gen and Cheng, 1997). The basic idea is to maintain a population of candidate solutions that evolves under a selective pressure that favours better solutions.

Generally, a GA is an iterative procedure that operates on a finite population of N chromosomes (solutions). The chromosomes are fixed strings with binary values (0 or 1 called alleles) at each position (or locus). Each chromosome of the population is evaluated according to a fitness function. Members of the population are selectively interbred in pairs to produce offspring (new solutions). Genetic operators are used to facilitate the breeding process that results in offspring inheriting properties from their parents. The offspring are evaluated and placed in the population, possibly replacing the weaker members of the last generation. Thus, the search mechanism consists of three phases: *evaluation* of the fitness of each chromosome, *selection* of the parent chromosomes, and *application genetic operators* to the parent chromosomes. The new chromosomes resulting from these operations form the population for the next generation and the process is repeated until the system ceases to improve.

The theoretical basis for the genetic algorithm is the *Schemata Theorem* (Holland, 1992, Michalewicz, 1996, Gen and Cheng, 1997), which states that individual chromosomes with good, short, low-order schemata or building blocks (i.e. beneficial parts of the chromosome) receive an exponentially increasing number of trials in successive generations.

2.2.1.1 The General GA Algorithm

```
Procedure GeneticAlgorithm
Start
//start with an initial generation
G = 0;
//initialise a random generation of fixed-format strings
Pop = InitPopulation (G);
//evaluate the fitness of individuals in the population
evaluate (G);

repeat
    //increase generation counter
    G=G+1;
     //generate new population using fitness-proportionate
     reproduction
     Pop1 = select (Pop);
     //crossover genes
     Pop1 = crossover (Pop1);
     //mutate genes
     Pop1 = mutate (Pop1);
     //evaluate fitnesses of new population
     evaluate (Pop1);
     //replace population with new generation
     Pop = Pop1;
until StopCriterion
end
```

There are various alternatives and modifications of this algorithm but the essential structure is normally similar (Lee and Takagi, 1993, Hsu, *et. al.*, 1996, Punch, *et. al.*, 1993, Liepins, *et. al.*, 1987). One common change is to incorporate the reproduction operation into the crossover and mutation operations - individuals are selected fitness-proportionately, crossed over (or mutated) and inserted into the new generation in a single operation.

2.2.1.2 Some Example Applications of GA to Engineering Problems

- Dereli *et. al.* (1998) used GA to determine the best possible positions of cutting tools on the turret or magazine of CNC machine tools to be used for machining components in order to achieve the optimality of process plans. Another work

from a similar domain is Mizugaki *et. al.*'s (1994) paper. They applied GA to milling tool selection problem.

- GA has been applied to complex production scheduling-rescheduling and sequencing problems by Fang, *et. al.* (1995), Jain and Elmaraghy (1997), Murata, *et. al.* (1996-a,b).

- Chipperfield and Fleming (1996) applied GAs to the design of a multivariable control system for a gas turbine engine. Pearce and Cowley (1996) used a GA to calibrate a gas turbine engine.

- Balakrishan and Jacob (1996) developed a GA for product design problems. They also concluded that GA is superior to Dynamic Programming procedures according to their application.

- Ettl and Schwehm (1994) developed a GA based design methodology for Kanban controlled production lines using queuing networks.

- Stockton and Quinn (1995) presented the application of GAs to aggregate production-planning problems. Their work incorporates many aspects of aggregate production planning. They concluded that GA can be applied to this problem. Husbands *et. al.* (1995) also studied production planning problems by using GAs.

- Vancza and Markus (1991) discussed the flaws of classical planning methods and outlined a new approach by which a large portion of domain-related knowledge can be represented and passed to a learning method using genetic algorithms.

- Suresh *et. al.* (1995) applied GAs to facility layout problems. They represented a GA for solving the quadratic assignment problem formulation of the facility layout problem. Kazerooni *et. al.* (1996) presented a GA based integrated approach for solving cellular manufacturing layout problems. Gupta *et. al.* (1996) also developed a GA based approach to cell composition and layout design problems.

- Homafair *et. al.* (1995) presented an example application of GA to the famous travelling salesman problem. They concluded that performance of GA is highly dependent on representation and choice of neighbourhood operators for the travelling salesman problems.

- There are several applications of GAs to manufacturing cell formation problems. Joines *et. al.* (1996) developed an integer programming model for cell formation problems and applied GAs for its solution. Hwang and Sun (1996) developed a GA based heuristic procedure for cell formation problems. Venugobal and Narendran (1992), Hon and Chi (1994), Hsu and Su (1998) have also presented GA based cell formation strategies.

Due to its adaptive and problem independent nature, there are also current efforts to apply GA to multiple objective optimisation and simulation optimisation problems. This issue and related literature will be reviewed later in this chapter.

2.2.2 Tabu Search (TS)

Tabu search is a heuristic problem-independent optimisation method. It was first suggested by Glover (1986) and since then has become increasingly used. The basic idea of the method, described by Glover (1990,1993), is to explore the *search space* of all feasible solutions by a sequence of moves. A move from one solution to another is the best available. However, to escape from locally optimal but not globally optimal solutions and prevent cycling, some moves, at one particular iteration, are classified as forbidden or tabu. Tabu moves are based on the short-term and long-term history of the sequence of moves. A simple implementation, for example, might classify a move as tabu if the reverse move has been made recently or frequently. Sometimes, when it is deemed favourable, a tabu move can be overridden. Such aspiration criteria might include the case which, by forgetting that a move is tabu, leads to a solution which is the best obtained so far.

Suppose that f is a real valued objective function on a search space S, and it is required to find a $h \in S$ such that $f(h)$ has maximal value. For NP-complete problems, this requirement needs to be relaxed for finding a $h \in S$ such that $f(h)$ is close to the maximal value Glover (1990). This is because any known algorithm to determine the maximal solution requires time that is exponential in the problem size. Sub-optimal problems may be solved by halting when a certain threshold for an acceptable solution has been found or when a certain number of iterations has been completed.

A characterisation of the search space S for which TS can be applied is that there is a set of k moves $Q=\{q_1,q_2,....,q_k\}$ and the application of the moves to a feasible solution $s \in S$ leads to k usually distinct, solutions $Q(s)=\{q_1(m),q_2(m),.. ..,q_k(m)\}$. The subset $N(s) \subseteq Q(s)$ of feasible solutions is known as the neighbourhood of s. The method commences with a (generally random) solution $s_0 \in S$ and determines a sequence of solutions $s_0, s_1, s_2, ...,s_n \in S$. At each iteration, s_{j+1} is selected from the neighbourhood $N(s_j)$. The process of selection is first to determine the tabu set $T(s_j) \subseteq N(s_j)$ of neighbours of s_j and the aspirant set $A(s_j) \subseteq T(s_j)$ of tabu neighbours. Then s_{j+1} is the neighbour of s_j which is either an aspirant or not tabu and for which $f(s_{j+1})$ is maximal; that is $f(s_{j+1}) \geq f(s^*) \quad \forall \quad s^* \in (N(s_j)-T(s_j)) \cup A(s_j)$.

2.2.2.1 The General TS Algorithm

```
Procedure TabuSearch
Start
K=1;
Generate initial solution s;
Repeat
      Identify N(s) ⊆S (Neighbourhood set);
      Identify T(s) ⊆N(s) (Tabu set);
      Identify A(s) ⊆T(s) (Aspirant set);
      Choose s*∈(N(s)-T(s))∪A(s), for which f(s*) is maximal;
      s=s*;
      k=k+1;
until StopCriterion
end
```

Note that it is possible to avoid convergence at a local maximum, that $f(s_{j+1}) < f(s_j)$. The conditions for a neighbour to be tabu or an aspirant will be problem specific. For example, a move may be tabu if it could lead to a solution which has already been considered in the last m iterations or which has been repeated many times before. A

tabu move satisfies the aspiration criteria if, for example, the value of $f(s^*)$ with $s^* \in T(s_j)$ satisfies $f(s^*) > f(s_i) \quad \forall i, \ 0 \le i \le j$.

2.2.2.2 Some Example Applications of TS to Engineering Problems

TS has been applied to many diverse engineering problems. However, its literature is not as rich as GAs because it is a relatively new technique, although there is a growing interest in the research community. Example applications include:

- Islam and Eskioglu (1997) developed a TS algorithm to solve the single machine mean tardiness problem. They also compared the TS based technique with three other methods (i.e. Simulated Annealing and two heuristic algorithms) and concluded that TS provides a better solution than the other three approaches. Kato *et. al.* (1997) proposed an effective neighbourhood for the minimisation of mean tardiness of job shops by using TS. Taillard (1994), Laguna *et. al.* (1991) also applied the TS to scheduling problems.

- Al-Fawzan and Al-Sultan (1998) applied TS to determine the production rate, period batch size and production sequence when production rate, set-up cost and unit processing cost are sequence dependent in a production planning problem.

- Siarry and Berthiau (1997) applied TS to optimise multi-modal functions with continuous variables. They concluded that TS could successfully find optimum solutions.

- Bland and Dawson (1991) applied TS to design optimisation problems successfully. They solved electronic-circuit design problems with TS based algorithms.

- Hertz and Werra (1987) developed TS algorithms to solve graph-coloring problems. Hertz (1991) also applied TS to large-scale time tabling problems.

- Sun *et. al.* (1995) proposed a TS algorithm for the manufacturing cell formation problem. They modelled the problem as a graph-partitioning problem. Their algorithm solves this problem through improving cell configuration using the TS technique.

2.2.3 Simulated Annealing (SA)

Simulated annealing is a problem independent random search procedure that was initially proposed by Kirkpatrick *et. al.* (1983). The basic difference between GA, TS and SA is that GA and TS always work with a population of solutions while SA works on one solution at a time.

Annealing is the physical process of heating up a solid above its recrystallisation temperature, followed by cooling it down until it crystallises into a state with the desired lattice structure. During this process, the free energy of the solid is minimised. Practice shows that the cooling must be done carefully in order not to get trapped in locally optimal lattice structures with crystal imperfections.

In combinatorial optimisation, it is possible to define a similar process (Rutenbar, 1989). This process can be formulated as the problem of finding among a potentially very large number of solutions, a solution with minimal cost. Now, by establishing a correspondence between the cost function and the free energy, and between the solutions and the physical states, One can introduce a solution method in the field of combinatorial optimisation based on the simulation of physical annealing processes. The resulting method is called *simulated annealing*.

The physical annealing process modelled by using computer simulation is based on Monte Carlo techniques as follows; Given a current state i of the solid energy E_i, then a subsequent state j is generated by a perturbation mechanism which transforms the current state into a next state by a small distortion. The energy of the next state is E_j. If the energy difference, E_j-E_i, is less than or equal to 0, the state j is accepted as the current state. Otherwise it is accepted with a certain probability which is equal to $exp((E_i-E_j)/T)$, T is temperature. This probability is known as the *acceptance criterion*. This algorithm known as *metropolis algorithm* and is the basis of SA algorithm for optimisation problems.

The design of SA depends on three key concepts. The first is referred to as the *temperature* (or control parameter) and is essentially the parameter that controls the probability that a cost increasing solution will be accepted (for a minimisation problem). During the course of SA the temperature will be reduced periodically, reducing the probability of accepting a cost-increasing solution. The second key concept is *equilibrium*, or a condition in which it is unlikely that further significant changes in the solution will occur with additional sampling. For example, if a large

number of interchanges have been attempted at a given temperature without finding a better solution, it is unlikely that additional sampling will be productive. The third key concept is the *annealing schedule*, which defines the set of temperatures to be used and how many interchanges to consider (or accept) before reducing the temperature. If there are too few temperatures or not enough interchanges are attempted at each temperature, there is a great likelihood of stopping with a sub-optimal solution.

2.2.3.1 The General SA Algorithm

The main steps of SA are; initial solution, generation of a neighbourhood solution, acceptance/rejection of generated solution, and termination.

Assume that *(S,f)* is an instant of an optimisation problem and i and j are two solutions with costs $f(i)$ and $f(j)$, respectively. Then the acceptance criterion determines whether j is accepted from i by applying the following acceptance probability:

$$P_c\{accept\ j\} = \begin{cases} 1 & if \quad f(j) <= f(i) \\ \exp\left(\dfrac{f(i)-f(j)}{c}\right) & if \quad f(j) > f(i) \end{cases} \qquad 2.1$$

$c \in R^+$ *denotes control parameter*(*temperature*).

If c_k denotes the value of control parameter, L_k the number of transitions generated at the kth iteration then a SA algorithm can be shown as follows;

```
Procedure SimulatedAnnealing
Start
      Initialise (i_start , c_0 , L_0);
      k=0;
      i=i_start;
      repeat
            for l=1 to L_k do
            start
                  generate (j from S_i);
                  if f(j)<=f(i) then i=j
                  else
                  if exp[(f(i)-f(j))/c_k]>random[0,1) then i=j
            end;
            k=k+1;
            CalculateLength (L_k);
            CalculateControl (c_k)(c_k+1=c_k*α);
      Until StopCriterion
End
```

As it is seen from the algorithm the probability of accepting a worse solution is determined by comparing the acceptance probability with a random number generated from a uniform distribution on the interval [0,1].

2.2.3.2 Some Example Applications of SA to Engineering Problems

- Chen and Srivastava (1994), Sofianopoulou (1997) proposed SA algorithms to solve manufacturing cell formation problems. Selim and Alsultan (1991) used SA for clustering problems.

- Connolly (1992) developed a comprehensive computer program to solve zero-one integer linear programming problems with SA based algorithms. Zhang and Wang (1993) proposed SA based algorithms to solve mixed-discrete non-linear optimisation problems. Lin *et. al.* (1993) developed SA based algorithms to solve NP-hard combinatorial optimisation problems. Goffe *et. al.* (1994) made an extensive study about global optimisation of statistical functions with SA.

- Wilhem and Ward (1987) developed a SA algorithm to solve quadratic assignment based formulation of facility layout problems. Liu *et. al.* (1994) also applied SA to facility layout problems.

- Gangadharan and Rajendran (1994) developed a SA heuristic for scheduling flow-shops with the twin-objective of minimising make-span and total flow time. Liu (1997) proposed a SA algorithm to minimise the mean flow time in flow-shop scheduling problems. Satake *et. al.* (1998) also proposed a SA based algorithm to minimise make-span in flow-shop scheduling problems. Raghu and Rajendran (1995) developed due date setting methodologies for job shops by employing SA.

- Elperin *et. al.* (1990) proposed Monte Carlo annealing procedures for machine design optimisation applications. They also presented an application for the cost optimisation of a speed reductor.

- Chen and Tsai (1996) developed an optimisation algorithm based on SA for the optimisation of cutting conditions in multi-pass turning operations.

- Marett and Wright (1996) applied SA to multiple objective optimisation problems. However, in their application they employed a secondary method for the evaluation of multiple objectives (i.e. weighting approach). The original SA algorithm was not extended. Similar approaches via applying other secondary methods (i.e. Game theory approach) were also proposed by Bennage and Dhingra (1995) for structural design problems.

Due to its problem independent nature, SA also applied to simulation optimisation problems by various researchers. This issue and related literature will be reviewed later in this chapter.

2.3 MULTIPLE OBJECTIVE DECISION MAKING

Many design problems require simultaneous optimisation of multiple, and in many cases conflicting, objectives. In the Operational Research literature these problems are known as Multiple Objective Optimisation (MOO) problems. Generally a MOO problem is of the following form:

$$\begin{aligned} &\min \text{ or } \max \ F(X) \\ &\text{such that;} \\ &X \in S = [X | X \in A^n, g_i(X) \le a_i,\ h_j(X) = b_j] \qquad i=1,2,\ldots,m,\ j=1,2,\ldots,n \end{aligned} \tag{2.2}$$

Where, X is an n-dimensional vector of the decision variables;

$F(X)=\{f_1(X), f_2(X), \ldots, f_k(X)\}$ is the set of objective functions; and S is the set of feasible solutions, bounded by m inequality constraints (g_i) and n equality (h_j) constraints, a_i and b_j are constants. For continuous variables $A=\Re$, for discrete variables A contains the set of permissible values.

2.3.1 Pareto Optimality

Pareto optimality is an economics term for describing a solution for multiple objectives (Ignizo, 1982). It is generally used to characterise optimal solutions to a MOO problem. The Pareto optimal (non-dominated) solution is defined as follows: a solution $X^* \in S$ is Pareto optimal if and only if there exists no $X \in S$ such that

$f_i(X) \leq f_i(X^*)$ for $i=1,2,3,...,k$ with $f_i(X) < f_i(X^*)$ for at least one value of i. In other words, the solution X^* is Pareto optimal if no objective function can be improved without worsening at least one other objective function. Figure 2.2 shows four geometric examples of Pareto optimality.

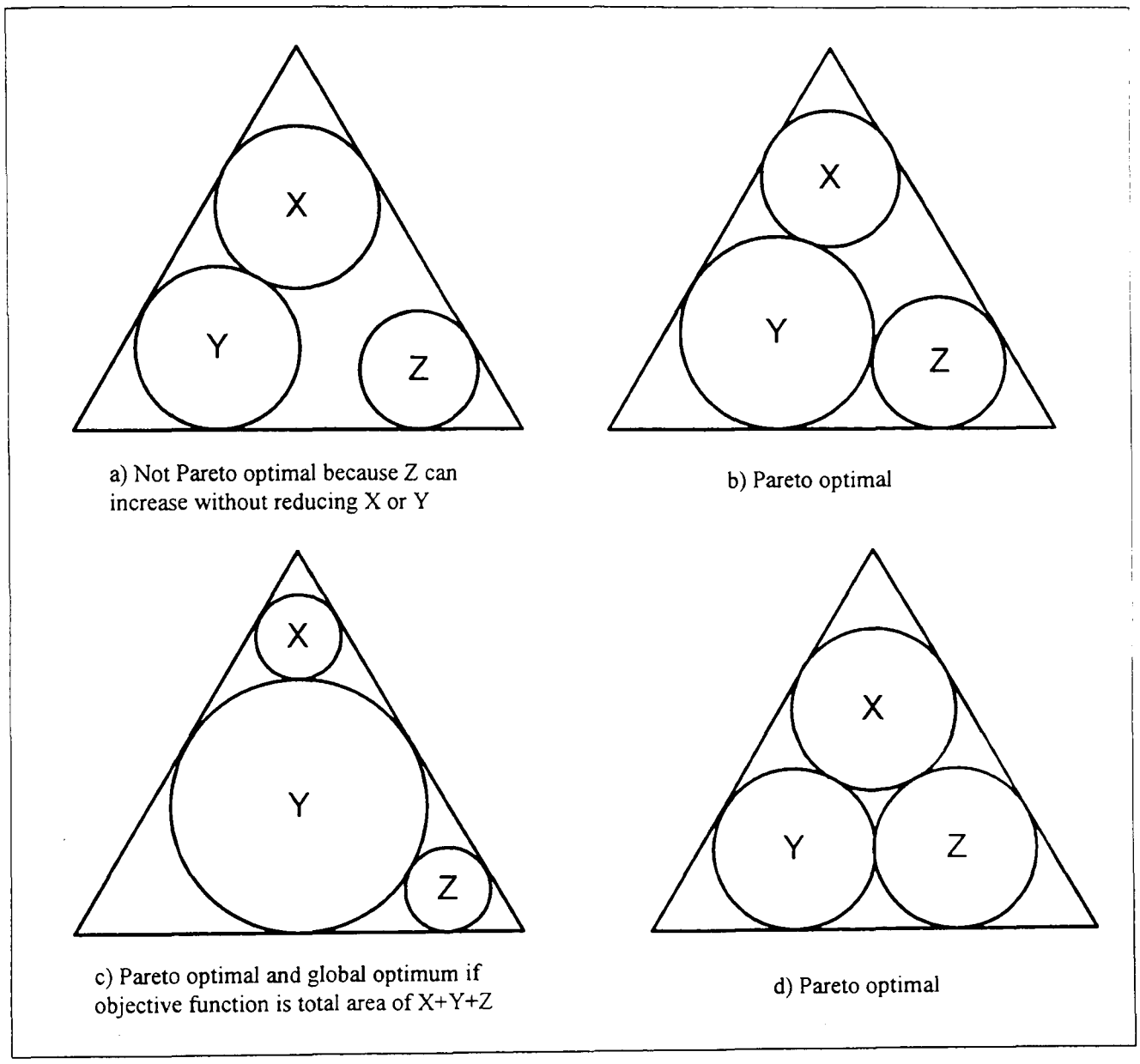


Figure 2. 2 Graphical explanation of Pareto optimality

In these figures, the circles represent *objectives* that are satisfied best when the area of the circle is maximised. The *constraints* are that the circles may not overlap and must fit within the triangle.

One might further impose a global objective function in this case that is equal to the sum of the circle areas. Only one of these figures is globally optimal (Figure 2.2-c) whereas three of them are Pareto optimal (Figure 2.2-*b,c,d*). One figure is not Pareto optimal because the area of one circle may be enlarged without violating the constraints (Figure 2.2-*a*).

Pareto optimality is a predicate. While one may be able to assign a quantitative metric, such as the area of the circle, the answer as to whether the global solution is Pareto optimal is "yes" or "no". It does not matter initially how much a circle can be enlarged, only that it can be. How much is to be evaluated after the possibility is noted. A corollary is that Pareto optimality does not address local extrema with respect to any utility. Neither does Pareto optimality provide a method for choosing among preferences or alternatives.

Nevertheless, tracking Pareto optimality is an important function. Detection of a lack of Pareto optimality is an alert to an opportunity to improve the design that otherwise might be missed, especially when no one expert understands all of the problem dependencies. Once such a lack has been detected, then special purpose algorithms can provide various evaluation functions that are likely to be domain-specific. Tracking Pareto optimality does not preclude such methods and it does not require an objective function that must compare "apples and oranges" in complex domains: it is a domain-independent function.

The set of Pareto optimal solutions usually consists of an infinite number of points and additional information is required to order the Pareto optimal set. This makes it

possible to bring additional considerations that are not included in the optimisation model, thus making the MOO approach a flexible technique for most design problems. Several techniques have been proposed for solving the MOO problem. Each method, in general, generates a different Pareto optimal solution that reflects the decision-maker's preference structure.

2.3.2 Techniques for Multiple Objective Optimisation

Many attempts have been made to find Pareto optimal solutions in MOO (e.g. Murata *et. al.*, 1996, Osyczka and Kundu, 1996, Dhingra and Lee, 1994, Chipperfield and Fleming, 1996, Gen, *et. al.*, 1997). The most commonly used techniques are:

- Utility function formulation (Weighting method)
- Global criterion formulation
- Goal attainment method
- Bounded objective function formulation (ε-constraint method)
- Game theoretical method
- The (lexicographic) constraint method
- Non-inferior set estimation method
- Genetic algorithms
- Goal programming method

The next few subsections discuss some of these techniques which are used to generate Pareto optimal solutions for the mathematical programming model given by Equation 2.2. Each of these techniques requires additional information from the decision-maker, and in general, generates a different Pareto optimal set.

In order to avoid working with different objectives in different units, the objective functions $f_i(X)$ are transformed into new objective functions (F_i) constructed as follows

$$F_i(X) = m_i f_i(X) \qquad i = 1,2,.....,k \tag{2.3}$$

$$m_1 f_1(X_0) = m_2 f_2(X_0) = ... = m_k f_k(X_0) = M \tag{2.4}$$

Here, the positive constant multipliers $m_1, m_2, ..., m_k$ are chosen so that at any feasible starting vector X_0 this scaling procedure ensures that all the objective functions are equal at a particular value of X_0. Hereafter, it will be assumed that k objective functions correspond to the k scaled objective functions given by equation 2.3. Further, it will be assumed that the MOO problem given by Equation 2.2 is non-convex, so that only locally Pareto optimal solutions are guaranteed. The non-convexity assumption holds for most practical engineering problems.

2.3.2.1 Utility Function Formulation

In the Utility Function formulation approach, the MOO problem given by Equation 2.2 is converted to

$$\begin{aligned} &\max \quad U(\vec{f}) \\ &Such \quad that; \\ &g_i(X) \le a_i \quad \forall i \\ &h_j(X) = b_j \quad \forall j \end{aligned} \tag{2.5}$$

where $U(\vec{f})$ is the utility function of multiple objective functions. The rationale for using $U(\vec{f})$ is that the decision-maker has some utility associated with each of the k objective functions. A utility function U can have many forms. The most common

form assumes that the decision-maker's utility function is additively separable with respect to all objective functions. Therefore, if $U_i(F_i)$ is the utility function corresponding to the objective function F_i, an overall utility function U is defined as

$$U(\vec{F}) = \sum_{i=1}^{k} U_i(F_i) \qquad 2.6$$

An optimum solution vector X^* is found by maximising the total utility function $U(\vec{F})$ (Equation 2.6) subject to the constraint set. A special form of Equation 2.6 that has been extensively used in MOO problems is given by

$$U = -\sum_{i=1}^{k} w_i F_i(X) \qquad 2.7$$

where w_i is a scalar weighting factor associated with the i^{th} objective function and indicates its relative importance. This additively separable form of the utility function (Equation 2.7) is also commonly referred as the "*Weighting method*", and serves as a sufficient condition for the calculation of Pareto optimal solutions (Ignizo, 1982).

The main advantage of the utility function formulation is its simplicity. It is easier to assess k unidimensional utility functions (U_i's) than to assess $U(\vec{F})$ directly. Similarly, it is easier to get w_i's from the decision-maker. The disadvantage of this approach is that, there are few cases where utility function is really additively separable, and w_i may depend not only on the achievement level of F_i but also upon the achievement level of F_i relative to F_j, for $i \neq j$. Further, if the problem is non-convex, this approach may miss all but a finite number of Pareto optimal solutions (Ignizo, 1982).

2.3.2.2 Global Criterion Formulation

This method belongs to a category of MOO techniques that require no articulation of preferences on the part of the decision-maker, once the problem objectives and constraints have been defined. This entails that the decision maker be willing to accept whatever solution is obtained by minimising some global criterion $F(X)$, for example, the sum of the squares of the relative deviations of the individual objective functions from the feasible ideal solutions. In other words, an optimum solution X^* is found by minimising

$$F(X) = \sum_{i=1}^{k} \left[(F_i(X) - F_i(X_i^*)) / F_i(X_i^*) \right]^p \qquad 2.8$$

Such that;

$g_i(X) \le a_i \quad \forall i$

$h_j(X) = b_j \quad \forall j$

The value of p corresponds to the utility function of the decision-maker and is usually taken as 2. X_i^* is the feasible ideal solution corresponding to the i^{th} objective function, and is obtained by minimising $F_i(X)$ with respect to the constraint set $X \in S$. For $1 \le p < \infty$, each solution obtained by solving Equation 2.8 is Pareto optimal, compromise solutions with $p = \infty$ correspond to a min-max criterion for which Pareto optimality is not guaranteed (Ignizo, 1982).

2.3.2.3 Goal Attainment Method

This method requires setting up goals $m_1, m_2, \ldots, m_k$ and weights $w_1, w_2, \ldots, w_k$ for the objective functions $F_1, F_2, \ldots, F_k$ respectively. The weights w_i relate the relative to under or over attainment of the desired goals (m_i). The following problem is solved to determine the optimal solution X^*

$$
\begin{aligned}
&\min \quad z \\
&Such \quad that; \\
&g_i(X) \le a_i \quad \forall i \\
&h_j(X) = b_j \quad \forall j \\
&F_i(X) - w_i z \le m_i \quad \forall i \\
&w_i \ge 0 \quad \forall i
\end{aligned}
\qquad 2.9
$$

where z is a scalar variable unrestricted in sign. The weights w_i are normalised so that

$$\sum_{i=1}^{k} w_i = k \qquad 2.10$$

In the case of under-attainment of the desired goals, a smaller weighting coefficient is associated with the more important objective functions. In the case of over attainment of the desired goals, a smaller weighting coefficient is associated with the less important objective functions. The optimum solution obtained using the goal attainment formulation is fairly sensitive to the goal vector $(\vec{m})$ and the weighting vector $(\vec{w})$ given by the decision-maker. Depending upon the prescribed values of the goal vector, it is possible that the weighting vector $(\vec{w})$ does not dictate the optimum solution at all. Instead, the optimum solution X^* is determined by the nearest non-dominated solution point from $(\vec{m})$. This may require that $(\vec{w})$ be varied parametrically to generate the entire set of Pareto optimal solutions. Further, if the goal vector is not chosen properly, there is no guarantee that the goal attainment formulation will terminate at a Pareto optimal solution.

2.3.2.4 Bounded Objective Function Formulation

In this method, the minimum and maximum acceptable achievement levels for each objective function F_i are specified by the decision-maker as L^i and U^i respectively. Then, an optimum solution X^* is found by solving the following problem

$$\begin{aligned} &\min \quad F_r(X) \\ &Such \quad that; \\ &g_i(X) \le a_i \quad \forall i \\ &h_j(X) = b_j \quad \forall j \\ &L^i \le F_i(X) \le U^i \quad \forall i \quad i \ne r \end{aligned} \qquad 2.11$$

This technique, also referred to as ε-constraint method (Ignizo, 1982), can be shown to lead to weak Pareto optimal solutions. However, if the optimal solution to the above problem is unique, then the resulting solution is Pareto optimal. Further, by systematically varying L^i and U^i, the bounded objective formulation can generate the entire set of Pareto optimal solutions for even non-convex problems. A difficulty with this method is to prescribe values for L^i and U^i prior to any preliminary solution. Since the designer has to specify these values in an information void, this may result in the mathematical programming problem given by Equation 2.11 having inconsistent constraints. Another question that needs to be addressed in this approach is which objective should be used for $F_r(X)$.

2.3.2.5 Game Theoretical Method

In this method, the MOO problem is viewed as a co-operative game theory problem involving several players, one corresponding to each of the objective functions. The system is assumed to be under the control of these intelligent adversaries, each

willing to compromise his (her) own objective in order to improve the overall solution. The basic approach can be summarised as follows

- Using X_0 as a starting point, solve k single objective optimisation problems given by

$$\begin{aligned} &\min \quad F_i(X) \\ &Such \quad that; \\ &g_i(X) \le a_i \quad \forall i \\ &h_j(X) = b_j \quad \forall j \end{aligned} \qquad 2.12$$

Let the optimum solutions be $X_i^*, i = 1,2,\ldots,k.$

- Construct a supercriterion or bargaining model S as

$$S = \prod_{i=1}^{k}\left[F_{iu} - F_i(X_w^*)\right] \qquad 2.13$$

where

$$F_{iu} = \max\left[F_i(X_j^*)\right] \quad i,j = 1,2,\ldots,k \qquad 2.14$$

and X_w^* represents the Pareto optimal solution obtained by solving the following problem:

$$\begin{aligned} &\min \quad F_w(w,X) = \sum_{i=1}^{k} w_i F_i(X) \\ &Such \quad that; \\ &g_i(X) \le a_i \quad \forall i \\ &h_j(X) = b_j \quad \forall j \\ &\sum_{i=1}^{k} w_i = 1 \\ &w_i \ge 0 \quad \forall i \end{aligned} \qquad 2.15$$

- Maximise the supercriterion and find the optimal convex combination $\bar{w}^*$ of the objective functions and the corresponding optimal solution to the problem, i.e. $X = X_w^*$. The game theory approach as presented above not only yields a Pareto

optimal solution, but also results in an optimum set of relative weights for the k objective functions.

2.3.2.6 Lexicographic (constraint) Method

In this method, the objectives are ranked in order of importance by the decision-maker. An optimum solution X^* is obtained by minimising the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives. The rationale for this method is that individuals tend to make decisions in this manner (Ignizo, 1982).

Let, the subscripts of the objectives denote not only the objective function number, but also the priority of the objective. The solution procedure is given as follows:

Step 1: Starting with X_0, minimise $F_1(X)$ subject to the constraint set. Let the resulting optimum solution be denoted as X_1^* and F_1^*.

Step 2: Starting from X_1^*, minimise $F_2(X)$ subject to the constraint set, and an additional constraint of the form $0.95F_1^* \le F_1(X) \le 1.05F_1^*$. Let the resulting solution be X_2^* and $F_2^* = F_2^*(X_2^*)$.

Step 3: Proceeding as outlined in Step 2, at the i^{th} stage the resulting problem is given as: Starting from X_{i-1}^*, minimise $F_i(X)$ subject to the constraint set, and i-1 additional constraint of the form $0.95F_j^* \le F_j(X) \le 1.05F_j^*$, $j = 1,2,\ldots,i-1$.

For a problem involving k criteria, there are a total of $k!$ ways in which the objective functions can be ranked by the decision-maker. Since the solution obtained using the

lexicographic method is fairly sensitive to the ranking of the objectives given by the decision-maker, one should exercise caution in applying this method when some objective functions are of nearly equal importance.

2.3.2.7 Genetic Algorithms

In order to extend GA to MOO problems, several approaches have been proposed. Almost all approaches that have been already proposed can be categorised into a population based non-Pareto approach or a Pareto based approach by their selection schemes (Foncesa and Fleming, 1993, Osyczka and Kundu, 1996, Gen, *et. al.*, 1997). In the following sub-sections, some of these algorithms are briefly explained.

2.3.2.7.1 Population based non-Pareto approach

The vector evaluated GA (VEGA) proposed by Schaffer (1985), is the first attempt to apply GA to MOO. It can be classified as a population-based approach because its selection procedure to form k sub-populations is implemented according to each of the k objectives separately. The outline of the VEGA can be written as follows:

```
Step 1: Initialisation
Step 2: Evaluation
Step 3: Selection to form k sub-populations using each of the k
        Objectives
Step 4: Genetic operations
Step 5: Elite strategy
Step 6: Termination test
```

Thus the VEGA has n search directions. Its search directions for the case of the two objective optimisation problem can be shown as in Figure 2.3. As it is expected from Figure 2.3, this approach can easily find the solutions A1 and A4, but it is not easy to find the solutions A2 and A3.

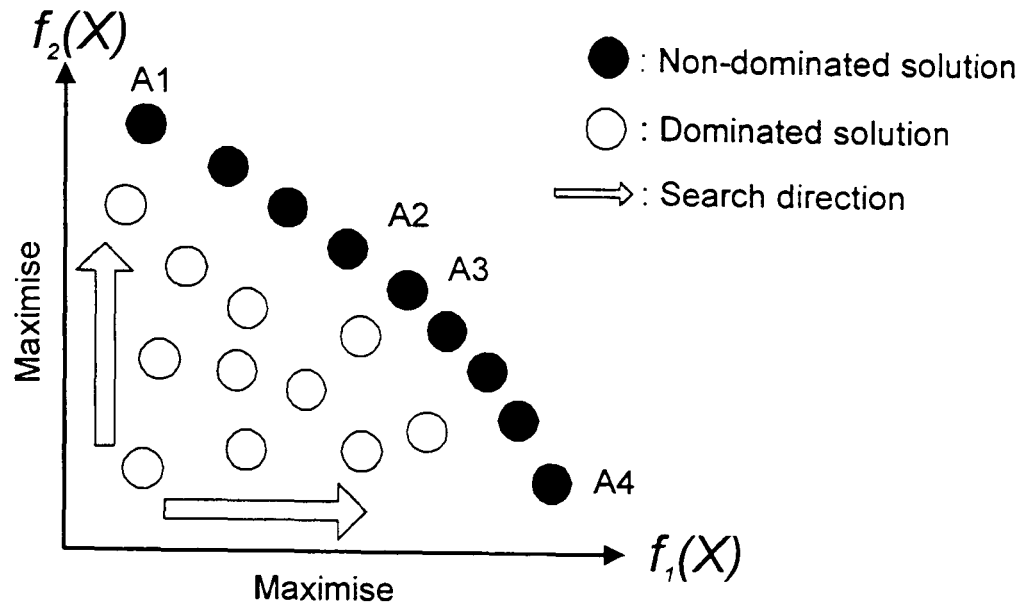


Figure 2. 3 The search directions in VEGA

2.3.2.7.2 Pareto based approach

Horn, Nafpliotis and Goldberg (1994) proposed the Niched Pareto GA (NPGA), which can be classified as a Pareto based approach. In NPGA, the Pareto domination tournament is employed as a selection procedure. Firstly, two candidates for selection are picked at random from the current population, and a comparison set consisting of a predefined number of individuals is also selected from the current population. Each of the candidates is then compared by at least one solution of the comparison set but the other is not dominated, the latter is selected for a crossover procedure. If neither or both are dominated by the comparison set, a fitness sharing technique is adopted. The outline of the NPGA is as follows:

```
Step 1: Initialisation
Step 2: Evaluation
Step 3: Selection
      Step 3.1: Select two candidate solutions from the current
                population, and select a certain number of
                solutions to form a comparison set.
      Step 3.2: Compare each candidate solution with the comparison
                set and determine a winner. If a single winner
                cannot be determined, go to Step 3.3. Otherwise,
                end this step.
      Step 3.3: Fitness sharing.
Step 4: Genetic operations
Step 5: Termination test
```

2.3.2.7.3 MOGA: Multi-objective genetic algorithm

MOGA is proposed by Murata and Ishibuchi (1995). It may be classified as a population based approach because a selection procedure based on the following weighted sum of the k objectives is performed to form a couple of solutions for a crossover procedure:

$$f(X) = w_1 f_1(X) + w_2 f_2(X) + ,\ldots\ldots, + w_k f_k(X) \qquad 2.16$$

where $w_1, w_2, \ldots, w_k$ are non-negative random weights for the k objectives, which satisfy $w_1 + w_2 + ,\ldots, + w_k = 1$. The search directions of the MOGA are shown in Figure 2.4. While the weighted sum approach tends to fail to find a non-convex Pareto front, this approach can keep all nondominated solutions found during the execution of the algorithm. This is because the algorithm separately keeps a tentative set of non-dominated solutions that are found during the execution of the algorithm. A pre-specified number of solutions in the tentative set of nondominated solutions are selected and added to the current population as elite solutions. The MOGA can be written as follows:

```
Step 1: Initialisation
Step 2: Evaluation
Step 3: Selection. Repeat the following procedure to select parent
        Solutions.
      Step 3.1: Randomly specify the weight values in equation 2.16
      Step 3.2: Select a pair of parent solutions according to the
                selection probability.
Step 4: Genetic operations
Step 5: Elite strategy
Step 6: Termination test
```

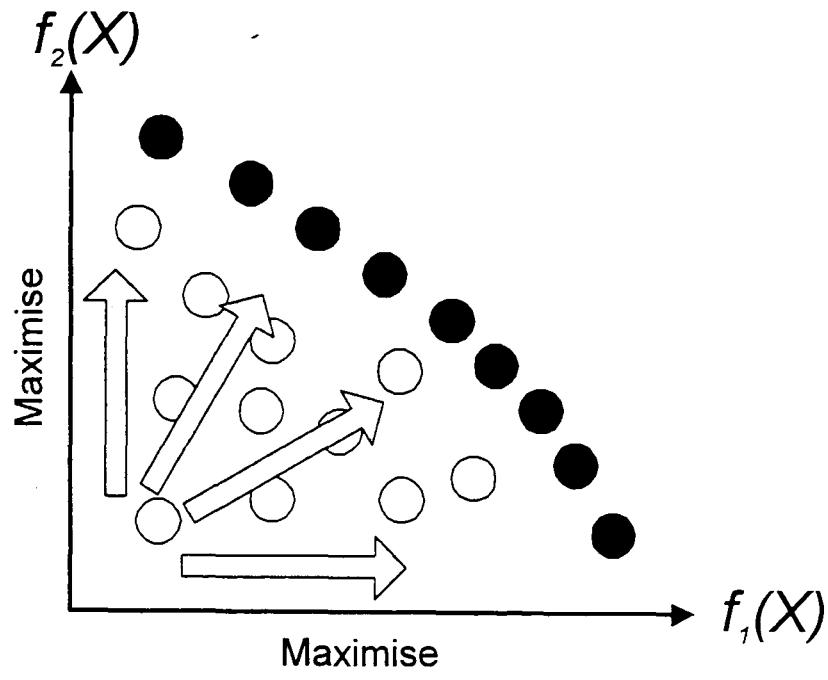


Figure 2. 4 Various search directions of MOGA

2.3.2.8 Goal Programming Method

In goal programming, there are two basic models; the Archimedian model and the Preemptive model. The Archimedian model deals with generation of candidate solutions whose criterion vectors are closest, in a weighted L_p metric sense to the utopian set in the criterion space. The preemptive model, on the other hand, generates solutions whose criterion vectors are most closely related in a lexicographic sense, to points in the utopian set. The Archimedian model is described in this section, the Peemptive model is explained in detail in the following sections and a TS algorithm is developed in this study for its solution (Chapter 5). It is also used as the main framework for solving MOO models through this study.

In the simplest version of Archimedian goal programming, a decision-maker sets goals and relative weights for each of the objective functions that he (she) wishes to attain. An optimum solution X^* is then defined as the one that minimises the

weighted sum of the deviations from the set goals. Thus, the goal programming formulation of a multiple objective problem leads to

$$
\begin{aligned}
&\min \quad \left[\sum_{j=1}^{k} w_j \left(d_j^+ + d_j^-\right)^p\right]^{1/p} \qquad p \geq 1 \\
&Such \quad that; \\
&g_i(X) \leq a_i \quad \forall i \\
&h_l(X) = b_l \quad \forall l \\
&F_j(X) - d_j^+ + d_j^- = m_j \quad \forall j \\
&d_j^+ \geq 0 \quad \forall j \\
&d_j^- \geq 0 \quad \forall j \\
&d_j^+ d_j^- = 0 \quad \forall j
\end{aligned}
\qquad 2.17
$$

Where, m_j are the goals set by the designer for the j^{th} objective function, and d_j^+ and d_j^- are the under and over achievement from the target goals for the j^{th} objective function. The value of p is based on a utility function chosen by the designer. If the goals m_j are set equal to F_j^* obtained by minimising individual objective functions F_j, it is not possible to obtain an over achievement of the goals m_j's. Consequently, the d_j^- need not to be defined. Thus the goal programming formulation given by Equation 2.17 reduces to

$$
\begin{aligned}
&\min \quad \left[\sum_{j=1}^{k} w_j \left(d_j^+\right)^p\right]^{1/p} \qquad p \geq 1 \\
&Such \quad that; \\
&g_i(X) \leq a_i \quad \forall i \\
&h_l(X) = b_l \quad \forall l \\
&d_j^+ = F_j(X) - F_j^*(X) \quad \forall j \\
&d_j^+ \geq 0 \quad \forall j
\end{aligned}
\qquad 2.18
$$

The goal constraints in the above formulation are soft constraints in the sense that they do not restrict the original feasible region S. In effect, they augment the feasible

region by casting S into higher dimensional space, thereby creating the augmented goal programming feasible region.

In their helicopter design research, Rao *et. al.* (1990) applied some of the above techniques (Global criterion formulation, Utility function formulation, Goal attainment method, Bounded objective function formulation, Game theory approach, Lexicographic method, Archimedian goal programming) and compared their relative efficiency. They concluded that none of the applied techniques could be considered superior to all other techniques in all circumstances. Murata and Ishibuchi (1997) compared the efficiency of GA based techniques in their flow-shop scheduling study. There was no significant difference between the efficiencies of the GA based techniques that are presented above.

2.4 MANUFACTURING SIMULATION AND SIMULATION OPTIMISATION

Simulation is simply the use of a computer model to *mimic* the behaviour of a complicated system and thereby gain insight into the performance of that system under a variety of circumstances (Thesen and Travis, 1991). In modelling and optimising manufacturing systems with structural and qualitative variables, simulation is sometimes the only possible method of evaluation. This is because the system configuration such as part routings, facility layout, queuing discipline, and many other factors cannot be quantified reasonably and thus, there does not exist an analytical expression of the objective function or the constraints for most of the design issues.

Manufacturing systems are generally modelled by applying discrete event simulation concepts (Shimizu, 1991, Norman and Norman, 1986, Swain and Farrington, 1994). A discrete-event simulation model is one in which system state changes only at a set of discrete points in time called event times (i.e. arrival of a part, finishing processing on a machine etc.). Between two successive event times, system-state does not change (Banks and Carson, 1986). There are many text-books where detailed information about discrete event simulation concepts can be found. For further detail refer to Pegden, *et. al.* (1995).

Performing a simulation analysis is generally not mathematically complex, but is requires a careful implementation of certain procedures from model building to the statistical analysis of outputs (Sargent, 1994, Cheng, 1993, Kelton, 1994, Charnes, 1993).

2.4.1 Methodology of a Simulation Study: A Short Review

Simulation is an extensively used method for reconfiguration, analysis, controlling of existing systems and design of new systems (Belmahdi and Nadif, 1995, Spedding, *et. al.*, 1997, Taboun and Bhole, 1993). Simulation is also a very valuable tool for educational purposes (Smith, 1989, Southern, 1979). It generally requires a complicated programming effort. However, a successful simulation study does not only consist of computer programming, but should also consider various other aspects in order to achieve its aims (Bakir, 1996).

The success of a simulation study is closely related to the achievement of the steps shown in Figure 2.5 (Prakash and Shannon, 1993, Saad, 1994). However, some

studies may not necessarily contain all the steps shown in Figure 2.5 in the order stated. Some studies may contain steps that are not included in the figure. Furthermore, a simulation study is not a strictly sequential process but often requires repeated iterations through several of the steps. For instance, if the real system does not yet exists, there is no real data to be collected (data acquisition step), and some appropriate estimates must be made, and then the subsequent model may be validated (Saad, 1994).

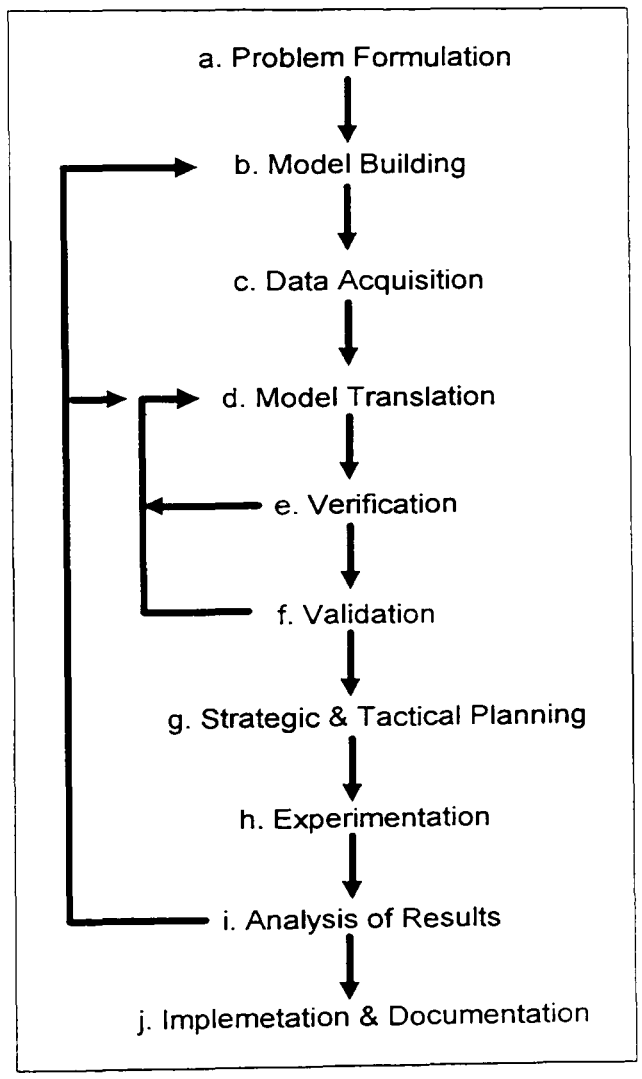


Figure 2.5 The simulation cycle (Prakash and Shannon, 1993)

a. Problem formulation

- State the objectives clearly
- Determine variables and constraints
- Determine the measures of performance

b. Model building

- Detailed study of the system to understand it for developing an appropriate model
- Preparing a flow chart of the model

c. Data acquisition

- Data should be collected to determine input parameters and probability distributions (e.g. part arrival time, transportation time etc.)
- Data should be collected to validate the model (e.g. average performance measures etc.)

d. Model translation

- Selection of an appropriate high level programming language or a simulation language.
- Preparation of programming code

e,f. Verification and validation

- Debugging
- Checking internal logic of the model to assure that each entity followed the correct logic and process
- Sensitivity analysis
- Comparison of simulation outputs with real data and other checks

g. Strategic and tactical planning

- What are the factors that affect the measures of performance?
- How many factors will be taken into account at one time?
- How many experiments will be required?
- How long each replication will take etc.?

h. Experimentation

- Determination of relationships between dependent and independent variables

- Comparison of different operating policies
- Evaluation of system behaviour
- Sensitivity analysis
- Optimisation etc.

i. Analysis of results

- Interpretation and presentation of results to the decision maker
- Determination of the best system design etc.

The statistical issues related to the simulation (i.e. determination of input probability distributions, analysis of simulation output etc.) can be easily analysed by the built in functions of many advanced simulation software packages, for example SIMAN-ARENA (Pedgen, *et. al.*, 1995).

2.4.2 Simulation with SIMAN

SIMAN is a simulation program for quickly and accurately implementing certain simulations on a computer. Its modelling framework is based on the system theoretic concepts developed by Zeigler (Pedgen and Ham, 1982). Within this framework, a fundamental distinction is stressed between the *system model* and the *experimental frame*. The system model defines the static and dynamic characteristics of the system. The experimental frame defines the experimental conditions under which the model is run to generate specific output data. For a given model, there can be many experimental frames resulting in many sets of output data. By separating the model structure and the experimental frame into two distinct elements, different simulation experiments can be performed by only changing the experimental frame, without changing the system model (Pegden and Ham, 1982). This characteristic of SIMAN

also gives the opportunity to easily parametrise a simulation model by using various experimental blocks.

The SIMAN software package used in this work divides the simulation process into three distinct activities:

- System model development
- Experimental frame development
- Data analysis

As shown in Figure 2.6 the SIMAN software consists of five individual processors which interact through four data files. The model processor is used to construct block diagram component models. The data file defining the block diagram generated by the processor is referred to as the model file. The experimental processor is used to define the experimental frame for the system model. The data file defining the experimental frame is referred as the experiment file. The link processor is used to combine the model and experiment file to produce the program file. The program file is input to the run processor that executes the simulation runs and writes the results to the output files. If an event is included in the system model, the user-written high level language subroutines are linked to the run processor before the simulation runs are executed. The output processor is used to analyse, format, and display the data contained in the output files.

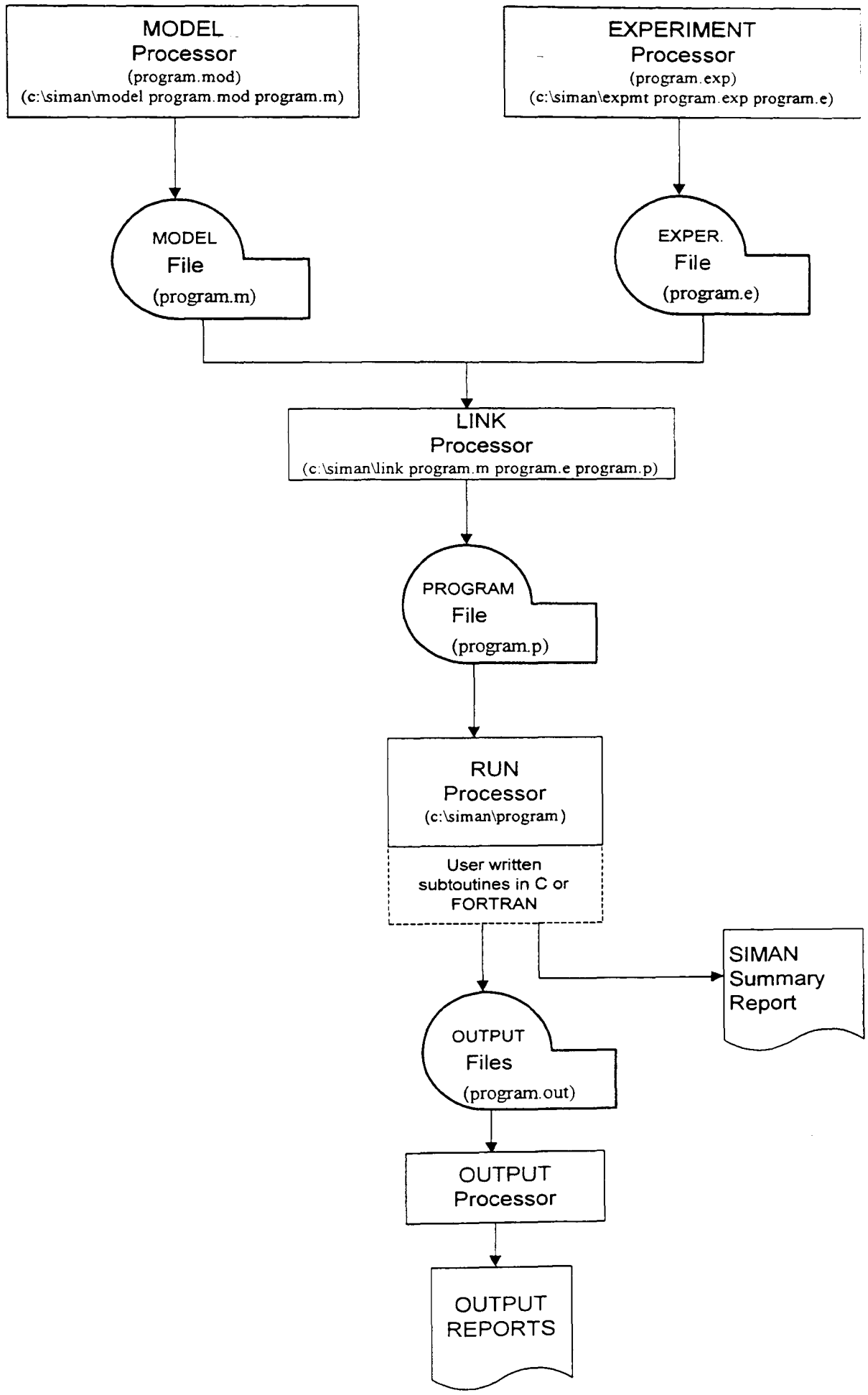


Figure 2.6 SIMAN software organisation (Pedgen and Ham, 1982)

To define a SIMAN program, blocks are defined and combined in different ways. Through these blocks *entities* move. Entities may represent such things as jobs, cars, or other items. Associated with *each* entity is a set of attributes, denoted *A(I)*,

A(2),....,A(n). Each attribute is a number, but can be interpreted in many different ways. For instance, one might set *A(1)* to be the time the entity entered the system, *A(2)* might be 0 if the entity represents a job of type X and 1 if it represents a job of type Y, *A(3)* is the service time of the entity, and so on.

These entities flow through a system of *blocks*. Blocks describe the actions an entity can take. In SIMAN, most blocks either modify an attribute, place entities in queues, or manipulate *resources*. Resources represent machines, workers, and other items that handle entities. There are many blocks in SIMAN and very complex problems can be modelled easily with them (Pedgen, *et. al.*, 1995). It is possible to input the necessary data into these blocks and run the simulation. However, BLOCKS and ELEMENTS are not necessary; it is also possible to create a SIMAN program with any ASCII word processor.

2.4.3 Simulation and Optimisation

In simulation models there are no functional relationships that can be manipulated by using techniques such as linear programming, geometric programming etc. to obtain the optimum combination values of decision variables. Therefore, a simulation model cannot be directly used for optimisation. However, it is possible to learn under what conditions a system performs most effectively and efficiently, by indirect use of a model for optimisation purposes through heuristic procedures.

Optimisation with a simulation model needs to use some kind of search technique. Since simulation itself is not an optimisation technique. It is necessary to combine

simulation with an optimisation procedure. It is possible to incorporate such methods for an optimum seeking procedure in a computer program.

The objective function in simulation models is expressed in terms of outputs of the models. In a sense a simulation model can be thought of as a stochastic objective function, whereby feasible input variables for the decision variables are converted into a value for the objective function. Because the objective function is not expressed in terms of the decision variables, optimisation techniques cannot be directly applied to the problem. There are two basic approaches to develop a technique to optimise the simulated systems:

- Direct search techniques
- Response surface methodology

The first category of approaches, such as pattern search (Clayton, *et. al.*, 1982), simplex method of Nelder and Mead (1965), combine an optimisation search procedure that does not require derivative information with a method for statistical comparison of two or more different systems. Clayton *et. al.* (1982) developed a modified pattern search procedure to optimise multi-response simulation models within a goal programming framework. They discussed the merit of their approach in detail. A military application is also mentioned in their paper (Clayton, *et. al.*, 1982). Another early application of simulation optimisation is the work done by Nolan *et. al.* (1972). They presented a recursive optimisation and simulation for the analysis of transportation systems. Garcia-Diaz *et. al.* (1981) proposed an integer programming based simulation optimisation strategy for solving the multi-machine interference problem. They applied various search procedures for solving the problem.

Mollaghasemi and Evans (1994), Teleb and Azadiar (1994) developed some heuristic direct search procedures to solve multiple objective simulation optimisation problems. However, as reported by Clayton *et. al.* (1982), these classical approaches have some disadvantages and limitations. One of the disadvantages is that these algorithms do not guarantee that a global optimal solution will be found and generally they produce local optimal solutions. A second disadvantage is that they are not efficient in the sense of requiring the fewest number of evaluations of simulations in order to achieve an optimum. Due to these known disadvantages, researchers in this particular area started to apply modern heuristic optimisation techniques for simulation optimisation. Haddock and Mittenthal (1992) applied simulated annealing for simulation optimisation. Their purpose was to investigate the feasibility of using a simulated annealing algorithm in conjunction with a simulation model to find the optimal parameter levels at which to operate a system. They applied their procedures to a small size hypothetical FMS system and demonstrated that the simulated annealing algorithm can result in an optimal or near-optimal solution to the problem. Lee and Iwata (1991) studied the problem of part ordering in a FMS through simulated annealing based simulation optimisation. Manz *et. al.* (1989) also applied simulated annealing to simulation optimisation. Their purpose was to find optimal parameter levels at which to operate a manufacturing system. Tautou and Pierreval (1995) applied genetic algorithms to simulation optimisation. Their specific application was to optimise configuration of a workshop producing plastic yoghurt pots by considering operational issues. Another application of genetic algorithms to simulation optimisation is the work done by Fujimoto *et. al.* (1995). They presented two heuristic approaches to a production-scheduling problem in FMSs. These approaches are integrated to seek efficiently the best combination of

dispatching rules in order to obtain an appropriate production schedule under specific performance measures. Sauer *et. al.* (1997) applied genetic algorithms to simulation optimisation. Their aim was to schedule FMSs. Joshi *et. al.* (1996), Bley and Wuttke (1997) also applied genetic algorithms to simulation optimisation problems. However, the majority of these studies are based on single objective function optimisation.

The second approach is to use the traditional response surface methodology (RSM). In this approach an approximating first and second order equation is fitted to the simulation response performance measure, using the series of simulation replications based on an appropriate experimental design. RSM has two primary disadvantages (Ferrel, *et. al.*, 1975). First, the RSM method is generally used with a simulation method to evaluate an objective function several times for each finite difference required along each factor or variable. The analyst configures the model, simulates with that configuration, and compares the alternative configuration outputs so as to improve the performance measure. This man-model interactive procedure requires many simulation runs and may lead to long computation times if each evaluation involves running a large simulation model. Secondly, there can be no guarantee that the results of an RSM procedure will always identify a truly optimal design. Ferrel *et. al.* (1975) reviewed and criticised the combination of simulation with various optimisation techniques. They stated that both the RSM and direct search methods have drawbacks, namely that the simulation results do not form an explicit mathematical objective function, and there is random variation in the output of simulation runs, and the number of computer runs must be limited. Smith (1976) developed a modular computer program to guide the optimum seeking solutions. He

concluded that his automated RSM program might be used for constrained and unconstrained optimum seeking in conjunction with deterministic or Monte Carlo simulations. Biles (1977) proposed a RSM-based multiple objective simulation optimisation approach by employing a non-linear goal programming framework.

In all the above simulation optimisation approaches, structural issues are not considered. Only the best possible level of variables and/or their combination (e.g. speed of transporters, set of dispatching rules etc) are estimated using a simulation model to optimise some performance measures. If structural changes are concerned (i.e. the composition of manufacturing cells, layout of the manufacturing system etc.) in the system under study then a unique simulation model is required for the evaluation of each configuration. Thus a simulation model generator is needed to automatically create simulation models in relation to the input provided by an optimum-seeking algorithm.

Based on the above review and observations, it is possible to divide simulation optimisation procedures into two broad categories:

- ***Non-parametric simulation optimisation*:** As noted above in this type of simulation optimisation approaches, only the level and/or combination of system variables are determined to optimise previously defined performance measure(s).
- ***Parametric simulation optimisation*:** In this type of simulation optimisation approach, structural changes are also considered. Therefore, a simulation model generator is required to generate and/or update the simulation models automatically during simulation optimisation.

The literature on parametric simulation optimisation is scarce. Ketcham and Watt (1989) proposed a simulation model generator for parametric simulation, by using spreadsheet like interfaces and a FORTRAN program that produces the simulation code. The work done by Malhotra and Mellichamp (1997) can be useful for parametric simulation optimisation studies. They developed a simulation code generator by using relational database management approaches. Morgan (1998) also briefly explained an automatic simulation model generator by using Microsoft Excel spreadsheets and the Visual Basic computer programming language. Such systems can be integrated with optimisation procedures to produce general-purpose parametric simulation optimisation software. A conference paper by Zhang and Azadivar (1997) presents an object-oriented approach for automatic generation of simulation models for a FMS. They also discussed the issue of using genetic algorithms to optimise simulation.

2.5 CELLULAR MANUFACTURING AND CELL FORMATION

In accordance with different production strategies (e.g. mass production, batch production, job shop production etc.) manufacturing systems were structured in several ways (i.e. functional, flow-line, cellular) (Kusiak, 1990). If a manufacturing environment can be characterised by high and stable demand with little product variety then the manufacturing system is normally laid down based on the operations sequences of products (i.e. flow-line) in order to obtain high efficiency. If a manufacturing environment can be characterised by low to medium and unstable demand with high product variety then a functional structure based on functionality of machines is preferred in order to increase the flexibility. After the introduction of the Group Technology (GT) concept, cellular manufacturing systems (CMS)

emerged. They are hybrid in nature and the main purpose is to bring together the advantages of previous approaches (Kusiak, 1987, 1990).

Several researchers have pointed out that the success of CMSs is mostly dependent on the success of their design (Shafer and Meredith, 1990). However, designing CMSs is known as a difficult problem (Kusiak and Chow, 1988, Zhou and Askin, 1998).

Several approaches have been proposed to design CMSs. A comprehensive review has been recently given by Reisman *et. al.* (1997). A classification of the strategies adopted is summarised as follows in this thesis:

a) ***Visual method***

b) ***Part Coding & Classification*** (Chang, *et. al,* 1991, Logar and Peklenik, 1991)

- Monocode
- Polycode
- Hybrid code

c) ***Production flow analysis*** (Burbidge, 1975)

- *Matrix formulation approaches*
  - * Similarity coefficient based methods (Seifoddini and Wolfe, 1986, 1994, Offodile and Grznar, 1997, Kerr and Balakrishnan, 1996)
  - * Matrix rearrangement methods (Chow and Kusiak, 1988)
- *Graph theory based approach* (Hadley, 1996, Lee and Garcia-Diaz, 1993)
- *Mathematical programming approaches*
  - * Integer programming (Gunasingh and Lashkari, 1989, Adil, *et. al.,* 1996, Boctor, 1991, Offodile, 1992, Srinivasan, 1990)
  - * Linear programming (Harhalakis, *et. al.,* 1994)
  - * Dynamic programming (Steudel and Ballakur, 1987)
  - * Multiple objectives and goal programming (Han and Ham, 1989, Wei and Gaither, 1990, Akturk and Balkose, 1996)
- *Mathematical programming & modern heuristic techniques* (i.e. SA, GA, TS) (Hon and Chi, 1994, Moon, *et. al.,* 1997, Hwang and Sun, 1996, Joines, *et. al.,* 1996, Chen and Srivastava, 1994, Sun, *et. al.,* 1995, Zhou and Askin, 1998, Hsu and Su, 1998)
- *Heuristic methods* (Al-Qattan, 1990, Khator and Irani, 1987, Sarker and Balan, 1996, Ballakur and Steudel, 1987, Purcheck, 1985)
- *Expert systems* (Basu, *et. al.,* 1989, Kusiak, 1990)
- *Neural networks* (Kamal and Burke, 1996, Chu, 1993, Moon, 1990)
- *Fuzzy set theory* (Gindy, *et. al.,* 1995,1996, Xu and Wang, 1989)
- *Pattern recognition* (Wu, *et. al.,* 1986)
- *Simulation* (Kamrani, *et. al.,* 1998)

d) ***Capability based approach***

- Fuzzy set theory (Gindy, *et. al.,* 1996)
- Mathematical programming & heuristic techniques (Baykasoglu and Gindy, 1999-c)

Visual method: In this method, part families are formed by an expert based on his (her) experience. It is not a systematic approach and cannot be applied if the number of parts are high (Kusiak, 1990).

Part coding: In this approach, every part in the system is coded alphanumerically in relation to their shapes, sizes and production features. Based on the part codes part families can be formed. Part coding is also very useful in design-retrieval processes and this is one of the main aims of this method (Gallagher and Knight, 1986). By visual and part coding methods only part families can be formed, machine cell formation is a secondary process. Visual and coding methods are also known as *part-oriented* methods (Wang and Roze, 1997). In part oriented methods, very much time, effort and money goes into accurate coding and the creation of an elaborate database which provides a weak connection between component features and machine tool grouping (Wang and Roze, 1997). For this reason, research on cell formation problems is mostly concentrated on the process based approach (i.e. production flow analysis).

Production flow analysis: In this method the machine route of every part is determined and transferred into a part/machine incidence matrix. This matrix provides the main data for the formation of part and machine cells. Cell formation techniques developed based on this strategy are called *production-oriented* methods (Wang and Roze, 1997). In production oriented methods only one route for each part is assigned. However, many times it is not possible to show the correct route of a part on an incidence matrix. This is because an entry in a part-machine incidence matrix only indicates whether a machine is used to process a part, not the number of times a

machine is needed and in which order machines need to be used (Cheng, *et. al.*, 1995). Moreover, there can be many possible alternative routes for each part and it is not easy to show these data on an incidence matrix. On the other hand, ignoring alternative routes can reduce the possibility of formation independent manufacturing cells and can cause excess machine duplication. Several researchers considered alternative process plans for the parts that prevent formation of independent cells by adding extra columns to the incidence matrix (Kusiak, 1990, Adil, *et. al.*, 1996). However, for a successful cell formation all possible alternatives should be evaluated. This is because alternative routes can have a big effect on the utilisation of the available capacity in the system.

Capability based approach: This approach is based on a capability/requirements analysis and included into the above classification after this research. This approach has the potential to overcome several shortcomings of the existing approaches. In this approach, capabilities of production resources and processing requirements of products are determined by using a common representation scheme known as Resource Elements (RE) (Gindy *et. al.*, 1996). Therefore, it is possible to determine the unique and overlapping capabilities of production resources and make use of this knowledge while forming manufacturing cells. Machining operations can also be used to define machine capabilities. However, using operations can increase the detail in problem modelling and solution requirements because there are hundreds of operations available in a typical job shop. This increases the solution space and therefore the problem size considerably. Additionally, operations are machine-specific and not easy to relate to capability representation (Gindy *et. al.*, 1996). REs can effectively define unique and overlapping capabilities of production resources

and processing requirements of parts. Employing REs can reduce the problem into a manageable size in cell formation applications (REs are explained in Appendix III). Figure 2.7 depicts the three representation schemes (i.e. machine-based, operation-based and RE based) that have been used in cell formation practices.

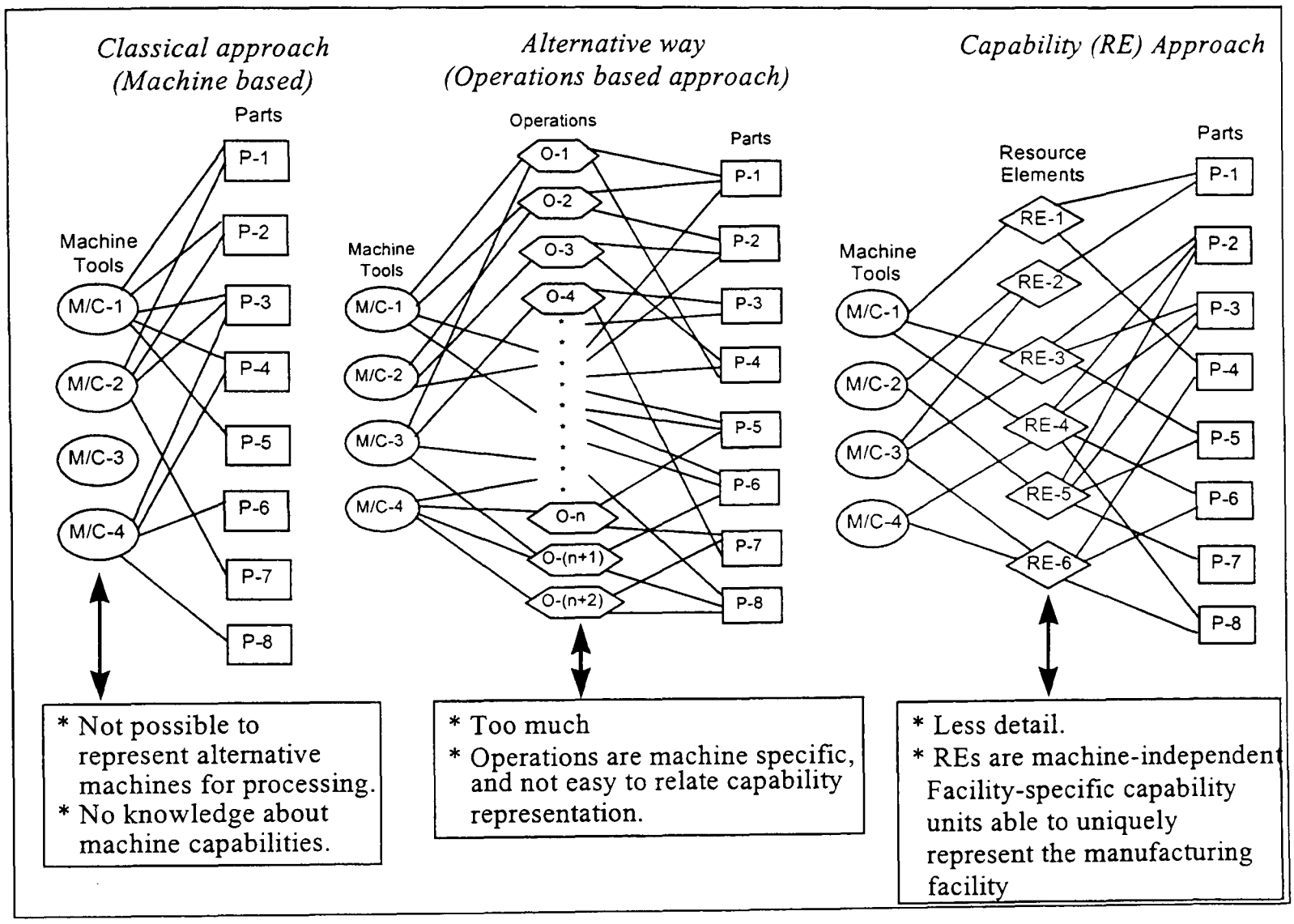


Figure 2.7 Three different approaches to relate part processing requirements to the available machine tools in cell formation applications

Based on the above approaches, there are three main strategies to form part-machine cells (Moon, 1990):

- Formation of machine cells first, then determination of part families.
- Formation of part families first, then determination of machine cells.
- Simultaneously forming part and machine cells.

In the literature, part family and machine cell formation problems are generally formulated separately. This increases the solution time and whichever problem is solved first can constrain the other in obtaining independent manufacturing cells. Therefore, it is advantageous to solve part-machine cell formation problem simultaneously (Moon, 1990). Moreover, many practical constraints have not been considered in many cell formation formulations (e.g. number of machines and parts in each cell, capacity constraints etc.). This is mainly due to the capability of the techniques employed for cell formation that do not allow consideration of these factors. Mathematical programming formulations are one of the most suitable alternatives for the cell formation problems because, they can integrate many important factors in the objective function and constraints (Sofianopoulou, 1997). However, powerful techniques are required for the solution of mathematical programming formulations. As discussed previously, modern heuristics techniques (i.e. TS, GA, SA) can be used for this purpose (Goldberg, 1989, Reeves, 1995).

2.6 LOADING AND SCHEDULING IN CELLULAR MANUFACTURING

Generally, CMSs are considered as dedicated manufacturing systems and they are not regarded as alternatives for dynamic manufacturing environments (McCarthy and Ridgway, 1995). This is because dividing the whole manufacturing shop into a number of cells reduces its flexibility. This can adversely affect the performance (Seifoddini and Djassemi, 1996, 1997). However, by employing effective loading strategies, the performance of CMSs facing changing production requirements can be sustained. Loading is a controlling issue of CMS. Greene and Sadowski (1980,1983) divided the control of CMS into two activities:

- Cell loading
- Cell scheduling

These two activities, which are a part of the planning and control of individual production units, lies at the very heart of the performance of manufacturing systems (Stoop and Wiers, 1996).

- *Cell loading is the determination of which cell (or cells), among the alternative cells the part will be assigned to.*

- *Cell scheduling on the other hand is the internal control of the jobs within each cell.*

In a cellular manufacturing environment loading and scheduling problems should be solved simultaneously to optimise the performance (Greene and Sadowski, 1980, 1983, Baykasoglu, *et. al.,* 1998-a).

2.6.1 Cell Loading

Loading of Flexible Manufacturing Systems is studied extensively in the literature Chen and Chung (1991), Kim (1993), Chen and Askin (1990), Stecke and Solberg (1981), Kusiak (1995), Wilson (1992). But, no serious attention has been given to the loading problems for the CMS. The CMS loading problem was first defined by Greene and Sadowski (1980,1983). They explained the dimensions and complexity of the problem and suggested some simple heuristics for CMS loading. They also performed simulation experiments under different operating conditions to test their

loading heuristics. They also developed a mixed-integer linear programming model for concurrently loading and scheduling cellular manufacturing systems (Greene and Sadowski, 1986). But the resulting formulation was not efficient even for a very small problem. Khator and Moodie (1979) developed a machine loading procedure by using part codes (Opitz code), they tried to develop a machine capability index which shows how well a machine can process a given part. They also carried out some simulation experiments on a hypothetical job shop to prove the superiority of their method. Elmaraghy and Gu (1989) proposed a feature-based part assignment method. In their work, machining requirements of part features were compared with machining capabilities (available operations) of existing manufacturing cells and parts were assigned to the cells which completely satisfied their processing requirements. The possibility of having inter-cell movement, operating conditions and performance measures of the system was not considered in their work. Choobineh (1984) developed a cell loading procedure. In his work, the existence of more than one feasible cell that can satisfy all processing requirements of each part is assumed. Then, each part is assigned to one of the candidate cells by solving a linear programming model with the objective of optimising total production costs. The possibility of having inter-cell movement was not considered in his work. Irastorza and Deane (1974) studied the loading problem for job shops with objectives of load balancing. They were only concerned with selecting new jobs to be released to the shop. Liang and Dutta (1990) also studied the loading problem of functional job shops. They expressed the importance of using alternative process plans and how these improved the shop performance. They developed a mixed-integer programming model for the loading problem with the objective of minimising total production cost. Tilsley and Lewis (1977), Ang and Willey (1984), Seifoddini and Djassemi (1996),

Albino and Garavelli (1997) proved that performance of a dedicated cellular manufacturing system can be considerably improved by assigning parts to alternative cells under dynamic manufacturing circumstances.

2.6.2 Cell Scheduling

Scheduling, by definition, is the determination of the order of the jobs onto each machine and the determination of the precise start time and completion time of each job on each machine. In reality, most viable control schemes do not perform cell scheduling but rather employ cell sequencing. Sequencing is limited to the determination of the order of the jobs onto each machine, and does not address timing (Sonmez and Baykasoglu, 1998).

In the CMS context, the control of each cell is nothing more than the scheduling of individual, small job shops and/or modified flow shops (Vaithianathan and Roberts, 1982, Greene and Sadowski, 1983). Therefore, most of the scheduling techniques developed for various types of systems can also be applied to CMS with minor or no modifications.

The scheduling or sequencing problem is formally stated as follows in the literature (Holthaus and Ziegler, 1997): n jobs are to be processed by m machines within a given time period in such a way that given objectives are optimised. Each job consists of a specific set of operations that have to be processed according to a given technical precedence order (process plan). An operation may require a deterministic or a stochastic processing time. If the precedence order of the operations is identical for all jobs the problem is called a flow shop scheduling problem in contrast to a job

shop scheduling problem where the jobs may have different sequences of operations (Yang, *et. al.,* 1994). If the set of jobs to be scheduled is available at the beginning of the scheduling process the problem is called static, whereas if the set of jobs to be processed is continuously changing over time the problem is called dynamic. In a deterministic problem all parameters are known with certainty. If at least one parameter is probabilistic the problem is called stochastic. The aim of the planning process is to find a schedule for processing all jobs optimising one or more goals (e.g. minimising mean tardiness, maximising throughput etc.) For efficient planning and controlling of production operations the detailed scheduling of all jobs has to be done in a short period of time, in real time at best.

In theory it is possible to determine optimal schedules for static sequencing problems. In practice the computation of optimal schedules for large problems is impossible since these problems belong to the class of NP hard problems (Pinedo, 1995). Therefore the existence of algorithms which are polynomial bounded in the problem size is very likely. The time requirements for calculating optimal processing of orders for a job shop scheduling problem occurring in practice would be beyond any scope of time. In recent decades many heuristic methods have been developed to solve deterministic scheduling problems (Pinedo, 1995). For scheduling dynamic and stochastic job shops a variety of dispatching rules has been proposed (Blackstone, *et. al.,* 1982, Ramaswamy and Joshi, 1994). The role of a dispatching rule is to select the next job to be processed from a set of jobs awaiting service. The dispatching rule selected can be very simple or extremely complex. Simulation models have been formulated to analyse their efficiency under various conditions and with respect to

different goals (Huang, 1984, Holthaus and Ziegler, 1997, Gindy and Saad, 1996, Brah, 1996).

There are many published papers addressing scheduling problems. In addition, there are thousands of industrial firms that have generated their own scheduling techniques of which there is little published knowledge. Pinedo (1995) provides an excellent textbook survey of the entire field of scheduling.

2.7 RECONFIGURATION ISSUES IN CELLULAR MANUFACTURING

2.7.1 Problems Associated with Cellular Manufacturing Systems

CMSs can be highly inefficient, since they are generally designed with a fixed set of part families in mind whose demand levels are assumed to be stable and their life cycles are considered to be sufficiently long. In fact, once a cell is formed, it is usually dedicated to a single part family with limited allowance for inter-cell flows. While such organisation may be adequate when part families are clearly identifiable and demand volumes are stable, they become inefficient in the presence of significant fluctuations in demand of existing products or with the frequent introduction of new ones. Wemmerlov and his colleagues (Wemmerlov and Hyer, 1987,1989, Wemmerlov and Johnson, 1997) made an extensive investigation of US industry and reported some of the problems. They concluded that there is a need to develop more effective cell formation techniques that produce more robust designs in order to lessen the deteriorating effect of changing production requirements. However, reconfiguration of CMS must also be considered to improve the performance of CMS, in addition to the advancements in cell formation procedures.

2.7.2 The Reconfiguration concept

Usually, facility layout studies result from changes that occur in the requirements for space, equipment and people. If requirements change frequently, then it is desirable to plan for change and to develop a flexible layout that can be modified, expanded, or reduced easily (Tompkins, *et. al.,* 1996). Flexibility can be achieved by utilising modular equipment, general-purpose production equipment and material handling devices etc. The change in the design of existing products, the processing sequences for existing products, quantities of production and associated schedules, and the structure of organisation and/or management philosophies (e.g. centralised, decentralised, hierarchical etc.) can lead to changes in layout. When these changes occur frequently, it is important for the layout to accommodate them. Such layouts are called flexible layouts (Abedzadeh and O'Brien, 1996). An important part of the response to change is the need to rearrange workstations or modify the system structure based on changing functions, volumes, technology, product mix and so on.

A re-layout problem arises when the location of an existing facility is a decision variable, i.e. in an existing facility a re-layout problem exists where the configuration and location of the existing facilities must be determined (Driscoll and Sawyer, 1985). With the introduction of new parts and changed demands, new part families and machine groups have to be identified. One possible way of minimising inter-cell movements is to relocate machines. Gupta and Seifoddini (1990) concluded that one-third of USA companies undergo major dislocation of production facilities every two years.

As mentioned above, over time the mix of parts, the volume of production for each part and the routing for each part in the system will change. If everything remains

constant for a long period of time, a dedicated set of facilities would be more appropriate but where this is not the case, there is a need to focus on a dynamic layout problem or a re-layout issue. The objective function in a dynamic layout problem is generally defined as the minimisation of flow costs plus rearrangement costs for a series of static layout problems (Rosenblatt, 1986, Lacksonen and Enscore, 1993, Barad and Aharonson, 1997, Driscoll and Sawyer, 1985). In a dynamic layout problem, rearrangement costs are added whenever an area contains different departments in consecutive time periods.

According to Lacksonen and Enscore (1993), the dynamic layout problem arises when we must balance the trade-off between the increased flow costs of inefficient layouts and added rearrangement costs. Afentakis, Millen and Solomon (1990) stated that when system characteristics (e.g. part mix) change, they can cause a significant increase in material handling requirements, consequently, it shows a need to consider re-layout. They defined two cost components for re-layout, cost of reconfiguration or relocation of equipment and cost of lost production. The cost of reconfiguration depends on the number of machines moved and/or the number of links in material handling changed.

Traditionally, the effectiveness of a layout has been connected to the flow of materials. Material handling cost is commonly used to evaluate alternative layout designs. The relative location of facilities in a functional layout has been determined under the criterion of material handling cost minimisation. Usually, the material handling cost is assumed to be an incremental linear function of the distances between the components of the system under study. Total estimated annual material

handling cost for a particular design is used to provide a quantitative measure of the flexibility of design. There are several papers available in the literature that studied the facility re-layout problems (Rosenblatt, 1986, Lacksonen and Enscore, 1993, Barad and Aharonson, 1997, Driscoll and Sawyer, 1985). However, in all these studies, functionally divided job shops were considered. Although, the inefficiency of cellular shops under changing environments has been criticised (Sasani, 1990), their reconfiguration has not been seriously investigated.

2.7.3 Reconfiguration from the Cellular Manufacturing Viewpoint

Rheault, *et. al.* (1995) defined the main characteristic features of today's dynamic manufacturing environments as follows: variable and stochastic demand, variable size of production lots, frequent and unpredictable changes in product mix, highly variable processing and set-up times, variable production sequences, and strong competition. As mentioned previously, the performance of CMS deteriorates when faced with changing production requirements (Sasani, 1990, Seifoddini and Djassemi, 1996). One of the reasons for performance deterioration in cellular manufacturing applications is the unsuitability of the existing configuration for changing production needs (Vakharia and Kaku, 1994).

To reduce the level of performance deterioration, reconfiguration of facilities should be considered. The reason for reconfiguration is clearly related to performance deterioration assessed by several performance measures i.e. utilisation levels, throughput, material handling cost etc. Unacceptable performance may necessitate reconfiguration. However, in functionally divided job shops only material handling cost is considered as the criteria for initiating the reconfiguration action. This might

be understandable, because in a functional shop functionally identical machines are collected in the same department and parts are routed from department to department based on their processing requirements. Therefore, those departments that have maximum load transfer between each other should be close. There is no reconfiguration action inside these departments therefore, only the relative locations of departments need to be changed. It is obvious that reconfiguration decisions are only related to material handling criteria for functional shops because, it is the only recognisable measure that can be improved by reconfiguration. On the other hand, the functioning logic of CMS is quite different. In a CMS material-handling cost is not the only measure that might trigger the reconfiguration. For example, in some production periods, there might not be inter-cell part transfer between cells so material handling cost is acceptable. But, due to demand fluctuations, a machine type in one cell might be heavily utilised, simultaneously in another cell an identical machine might not be utilised at all. In such cases, throughput levels decreases, tardiness and flow time increases, etc. Changing the cell membership of the idle machine can improve the overall performance of the shop. As can be seen from this simple example, in cellular shops many performance measures can be an indicator for reconfiguration. However, it is not easy to define a trade-off function for physically moving machines in a CMS because, estimation of the effects of moving machines between cells is generally not easy. For example, even if a machine has a very low utilisation in a cell, its removal to another cell might not improve the performance of CMS, due to detrimental effects of removing the machine from its original cell. After removal, another machine might become heavily utilised in the cell or it may not be possible to remove the machine without allowing inter-cell part

transfer due to unavailability of alternative capability in the cell. Additionally, under changing production conditions it is not easy to validate machine relocation costs.

Due to the complex nature of this problem, there is not a known simple strategy for the reconfiguration of CMS. Several researchers (McLean *et. al.*, 1982, Kusiak, 1990, Mohamed, 1996, Kochikar and Narendran, 1998, Montreuil, *et. al.,* 1992, Drolet, *et. al.,* 1990) proposed the logical cell concept as a mechanism for dynamically reconfiguring cellular manufacturing systems. In logical cell formation, there is no physical machine movement between cells, only the cell memberships of machines are updated. This eliminates the difficulties of evaluating trade-off between reconfiguration cost versus gains obtained from the reconfiguration. However, there is not a known procedure that relates the formation of logical cells to multiple performance measures.

2.8 CONCLUSIONS OF THE LITERATURE REVIEW

The following conclusions are drawn from the literature review:

- Classical optimisation procedures have shortcomings to solve complex engineering problems like cell formation, loading and reconfiguration.

- Although, there are some studies in the literature which show possible application of genetic algorithms to multiple objective optimisation, modern heuristic optimisation techniques have been generally applied to single objective optimisation problems.
- Classical optimisation procedures have several difficulties in solving simulation optimisation problems.

- Manufacturing cell formation problems are generally modelled as single objective optimisation problems without considering multiple objectives and some important constraints (capacities, demand etc.). Additionally, most of the models are only suitable for small problems.

- Although the loading problem of Flexible Manufacturing Systems is studied extensively, loading problem of CMS did not receive enough attention in the literature.

- Although the reconfiguration of functionally divided job shops has been studied by several researchers, reconfiguration of CMS has not received serious attention in the literature.

CHAPTER THREE

3. AN OVERVIEW OF THE PROPOSED FRAMEWORK

3.1 INTRODUCTION

This chapter presents an overview of the "Multiple objective decision support framework for configuring, loading and reconfiguring cellular manufacturing systems". The framework contains a number of interconnected modules and a variety of techniques are developed and used in each module. The main goal of the proposed framework is to guide reconfiguration decisions for CMS facing changing production requirements.

The life cycle of manufacturing cells starts with the initial cell formation process (*configuration module)*. The reconfiguration is performed if manufacturing cells cannot satisfy the required performance levels set by the decision-maker. Reconfiguration is achieved by generating virtual manufacturing cells[1]. The efficiency of the existing virtual cells is evaluated with the proposed loading system (*loading module*) at the beginning of each loading period. If the system performance measures are not satisfactory then the *reconfiguration module* generates new virtual cells.

In the proposed framework generic capability units which are known as Resource Elements (RE) (REs are explained in Appendix III) are used to define processing

[1] Logical grouping of manufacturing resources

capabilities of virtual cells and processing requirements of products. REs are capable of representing shared and unique capability boundaries between virtual cells and resources in each virtual cell. Therefore, *inherent flexibility*[2] can be better utilised while designing and controlling CMSs.

Cell formation, loading and reconfiguration decisions are made based on multiple objectives in the proposed framework. Tabu search based multiple objective optimisation algorithms are developed in order to evaluate multiple objectives.

This chapter gives a brief explanation about the framework and its modules. Detailed explanations about each module will be given in the subsequent chapters of the thesis.

3.2 AN OVERVIEW OF THE RECONFIGURATION PROBLEM IN CMS

In cellular manufacturing applications, cell formation is usually considered as a long-term decision. Various factors can be considered at the cell formation stage in order to create a robust CMS design. But due to frequent changes in production requirements in today's markets, performance deterioration is usually inevitable. Therefore, a decision-support mechanism is required to guide the reconfiguration actions to reduce or eliminate the detrimental effects of changing production requirements. The behaviour of a manufacturing system in accordance with changing production requirements can be better described schematically as shown in Figure 3.1.

[2] Unrealised potential of a manufacturing system in supplying alternative resources for part processing.

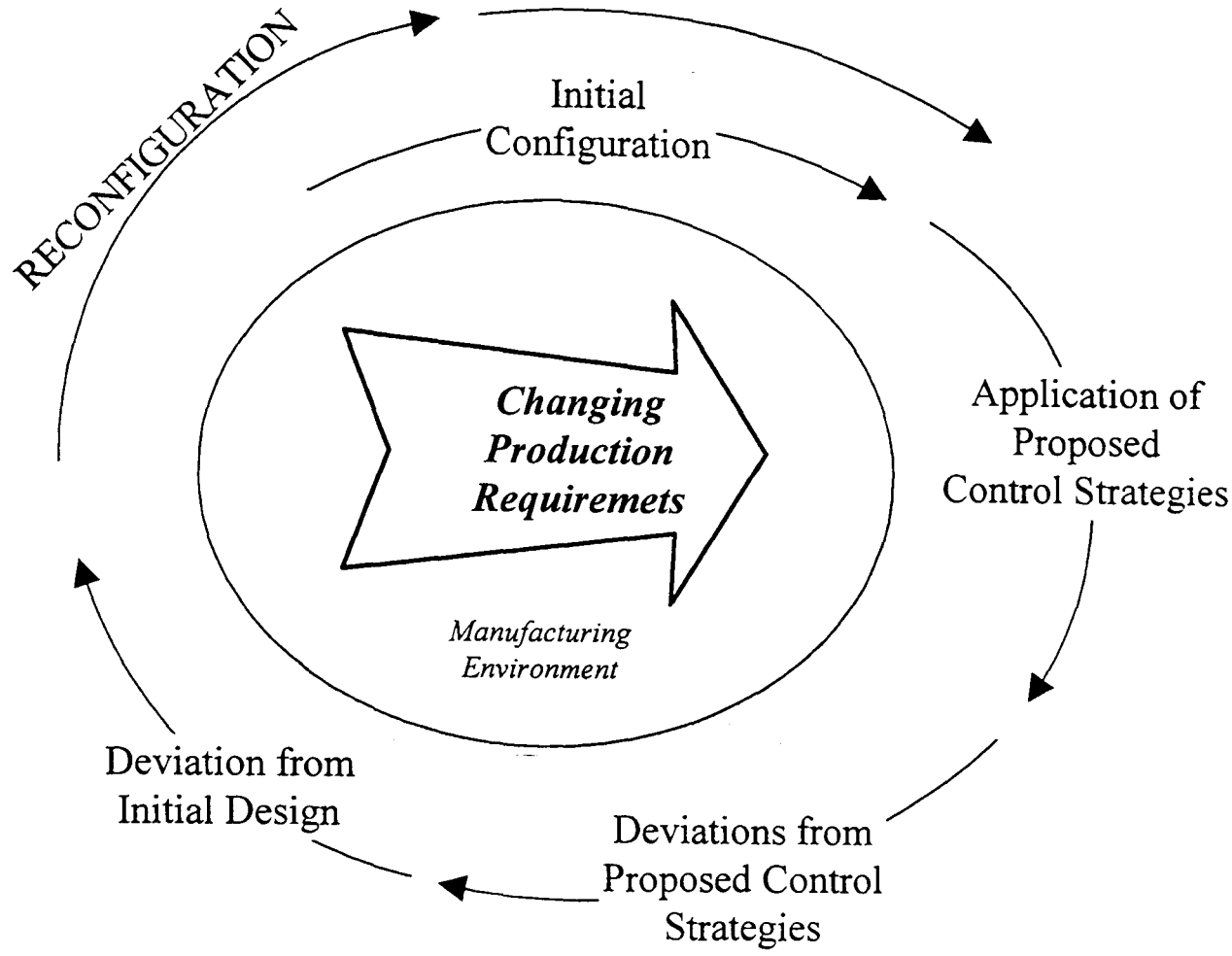


Figure 3.1 Changes in cellular manufacturing system design in relation to changing production requirements

The objectives of reconfiguration are to improve system performance measures to satisfy "market demand" and "management goals" by making better utilisation of resources. In order to achieve reconfiguration, there are two important questions to be answered, the first one is "***When to reconfigure?***" and the second one is "***How to reconfigure?***" It is not easy to estimate the effects that changing production requirement on CMS performance. Therefore, a simulation and optimisation system is necessary in order to analyse the effects of changing production requirements on the CMS performance and to optimise its performance and configuration. In the proposed framework, the interaction between process planning, loading, manufacturing system simulation and cell formation is considered. The interaction between process planning and loading through simulation can give an indication

about when to reconfigure CMS by analysing its performance. Reconfiguration is very similar to configuration problem in many ways. Solution of the configuration (factory design) problem leads to hard cell boundaries. On the other hand, the reconfiguration problem is logical i.e. its solution determines soft cell boundaries (virtual cells) to deal with changing production requirements. If requirements stay constant for long periods then virtual cells may lead to hard cell boundaries. The following section introduces and briefly explains the proposed integrated framework that can be used for reconfiguration purposes.

3.3 THE PROPOSED FRAMEWORK

The proposed framework is depicted in Figure 3.2. As discussed in Section 2.7.3 of Chapter 2, the virtual cell concept is used as a reconfiguration strategy in the proposed framework. Reconfiguration is achieved by generating virtual manufacturing cells (VC). ***A virtual cell is an objective driven logical grouping of manufacturing resources***. Its creation can be decided by a production planning system. If current cells cannot cope with the new production requirements then a new set of VCs is generated. The main difference between a VC and a classical GT cell is in the dynamic nature of the VCs; whereas the physical location and identity of traditional GT cell is fixed, the VC is not fixed and may vary with changing production requirements in each loading period if necessary. The VC concept as defined above is a powerful concept, which allows the flexible reconfiguration of shop floors in response to changing production requirements (Drolet *et. al.*, 1995). On the other hand, VCs may demand more effective production control and if not generated carefully may result increased material handling requirements.

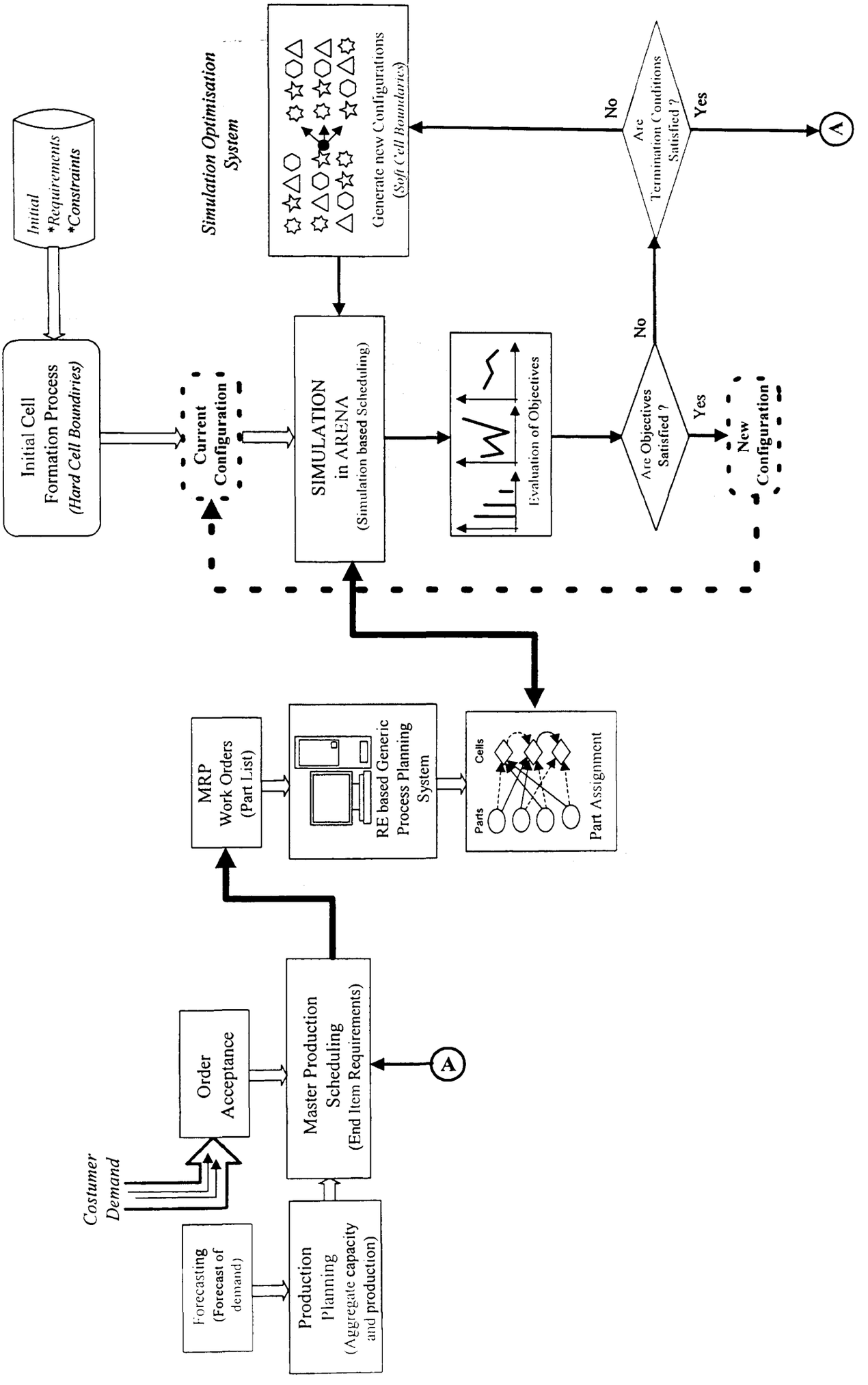


Figure 3.2 The proposed integrated framework for reconfiguring CMSs

To guide and achieve reconfiguration of CMS several decision making activities are interconnected in the proposed framework as shown in Figure 3.2. These activities include:

- Production planning
- RE based generic process planning
- Cell formation
- Cell loading
- Manufacturing system simulation
- Multiple objective optimisation
- Reconfiguration via virtual cell generation

Operation of the framework can be summarised as follows: The cell formation module generates initial hard boundary cells. In each production period a *part list* is produced by the production planning system which contains production requirements of parts in terms of Resource Elements. A multiple-objective Tabu search based simulation optimisation system is employed to load manufacturing cells by using the data provided by the part list. If previously defined performance indicators (total inter-cell traffic, mean tardiness, system utilisation level, throughput etc.) are unsatisfactory then the reconfiguration (creation of new virtual cells) is considered. Therefore, the *"When to reconfigure?"* question is answered here. The reconfiguration action is performed by a multiple-objective Tabu search based parametric simulation (refer to Chapter 2, section 2.4.3 for definition) optimisation model which generates possible virtual cell configurations and selects the one which best satisfies the desired performance levels (i.e. total inter-cell traffic, mean

tardiness, system utilisation level, throughput etc.). Therefore, the *"How to reconfigure?"* question is answered here.

In this study, cell formation (configuration), loading, multiple objective optimisation, parametric simulation and reconfiguration modules are developed. The availability of a production planning system (e.g. MRP-II) is assumed which can produce part lists for each production period. A generic process planning system (which can determine part processing requirements in terms of RE) and an RE-based scheduling system were previously developed in the Responsive Manufacturing Centre of the University of Nottingham (Gindy *et. al.*,1993, 1996). Each of the developed modules can also be used as stand-alone programs for their specific applications. Brief explanations about each module are given in the following sub-sections, detailed explanations with applications and comparisons are given in the subsequent chapters of this thesis.

3.3.1 Production Planning

Availability of an MRP-II type production planning system (PPS) is assumed. The interaction of the PPS with the proposed framework is shown in Figure 3.2. The PPS produces the end item requirements based on the demand for products in each production period. The output of PPS is a *part list* for each loading period. An example of an assumed part list is shown in Table 3.1.

Table 3.1 An example assumed part list

PART NAME (N0)	PROCESSING REQUIREMENTS (RE#)	PROCESSING TIME (min)	DEMAND	DUE DATE
PART-1	RE1-RE5-RE3	75-160-36	100	33days
PART-2	RE7-RE5-RE6-RE5	56-33-86-53	75	20
PART-3	RE1-RE5	97-194	50	17
PART-4	RE6-RE7	174-44	100	15
PART-5	RE1-RE2-RE5	85-85-67	80	22
...	...	...	...	...

3.3.2 Generic Process Planning

For each part in the part list, the output of the generic process planning system is a machine independent process plan expressed in terms of the REs needed for its execution. In this study 'part lists' are 'test part lists'. It is assumed that GENPLAN generated processing requirements of parts in the 'test part lists' in terms of REs. For detailed explanations about GENPLAN refer to Gindy *et. al.,* (1993). REs are explained in Appendix III.

3.3.3 Cell Configuration

Initial manufacturing cells are generated in this module. The capability-based multiple objective cell formation technique (MOCACEF 1.0) generates the initial cellular configuration. The basic logic in MOCACEF 1.0 is to match processing requirements of parts with the capabilities of available machines. Manufacturing cells are formed simultaneously by considering multiple objectives and constraints within a goal programming structure. The objective is to generate manufacturing cells with better utilisation levels, less machine duplication and higher flexibility. This module is explained in detail in Chapter 6 where some example applications along with a comparative study are presented.

3.3.4 Cell Loading

Loading of existing and/or reconfigured manufacturing cells is performed in this module. The Capabilities of the manufacturing cells and multiple performance measures are considered within a Tabu search based parametric-simulation optimisation framework that is formally represented as a goal programming model.

Detailed explanations about this module are given in Chapter 7 where some example applications are also presented.

3.3.5 Parametric Simulation System

A parametric simulation system is developed to obtain the performance of possible reconfigurations and loading scenarios. The general structure of the system is shown in Figure 3.3. When the loading or reconfiguration module generates a new solution, then the simulation model is automatically updated for the present solution vector by a specially developed C/C++ computer program (translator) to obtain its performance.

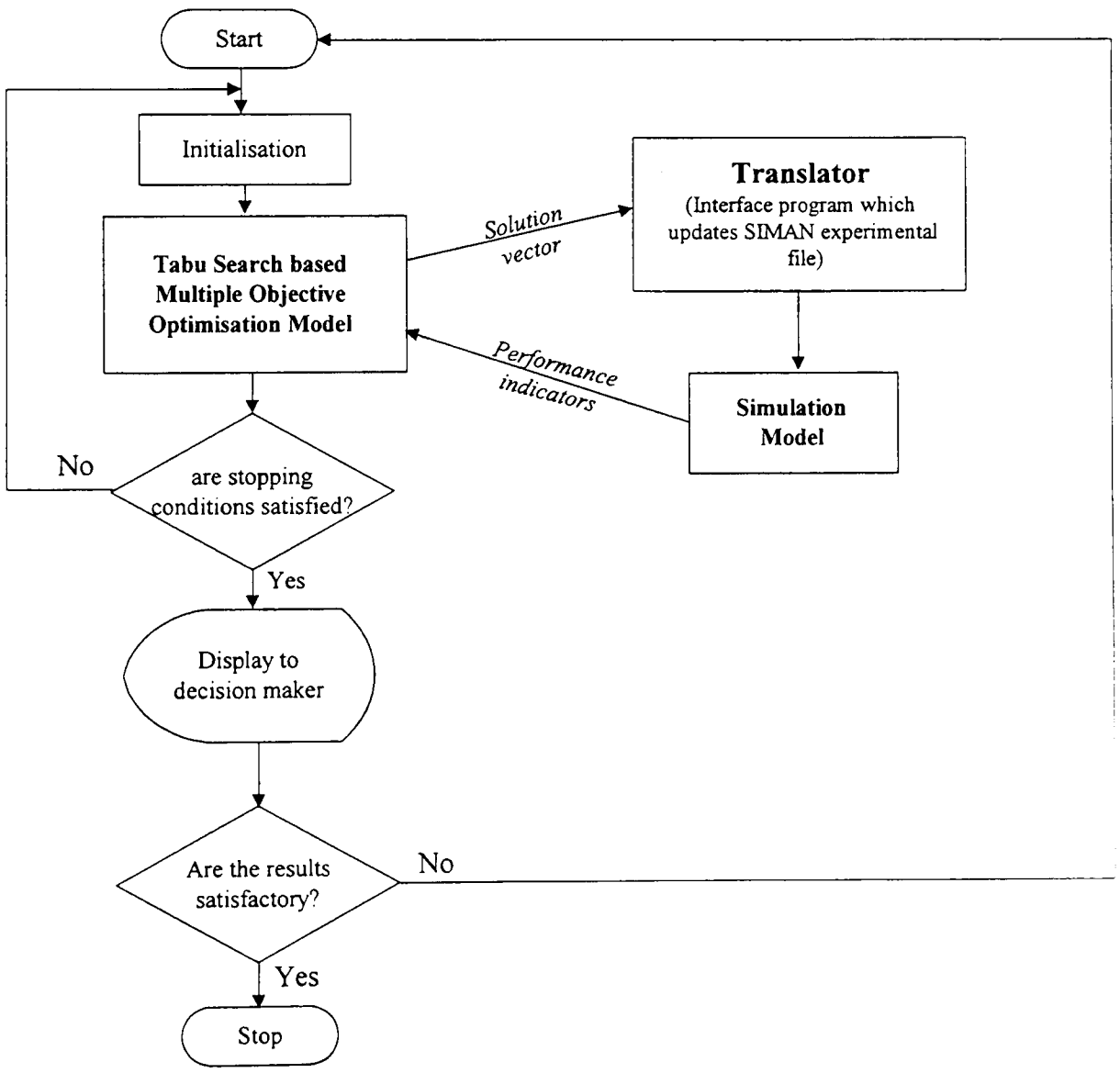


Figure 3.3 Parametric simulation optimisation system

The simulation module also determines the cell's schedule for the best solution obtained. The RE based simulation scheduling approach is proposed and extensively studied by Gindy and Saad (1996, 1997, 1998).

3.3.6 Multiple Objective Optimisation

Tabu search based general-purpose multiple objective optimisation algorithms are developed for solving the complex cell formation, loading and reconfiguration problems. In chapters 4 and 5 Tabu search algorithms are explained in detail.

3.3.7 Reconfiguration (Virtual Cell Formation)

The performance of the current configuration is evaluated by the simulation system for the coming production period and a decision about reconfiguration is made in this module. Based on the capabilities of the machines, and production requirements of products, candidate virtual cell scenarios which satisfy previously defined constraints (cell size etc.) are generated and the one which best satisfies the performance targets (total inter-cell traffic, mean tardiness, system utilisation level, throughput etc.) is adopted before actual production starts. The model is formally represented in a pre-emptive goal programming framework and solved by the multiple objective Tabu search algorithm. In the present approach a virtual cell configuration can be considered as a stage in which production system performance is satisfactory. If the system performance deteriorates due to changing production requirements, a new configuration that can satisfy the required performance levels is generated (see Figure 3.4). Detailed explanations on the reconfiguration module along with an application are given in Chapter 8.

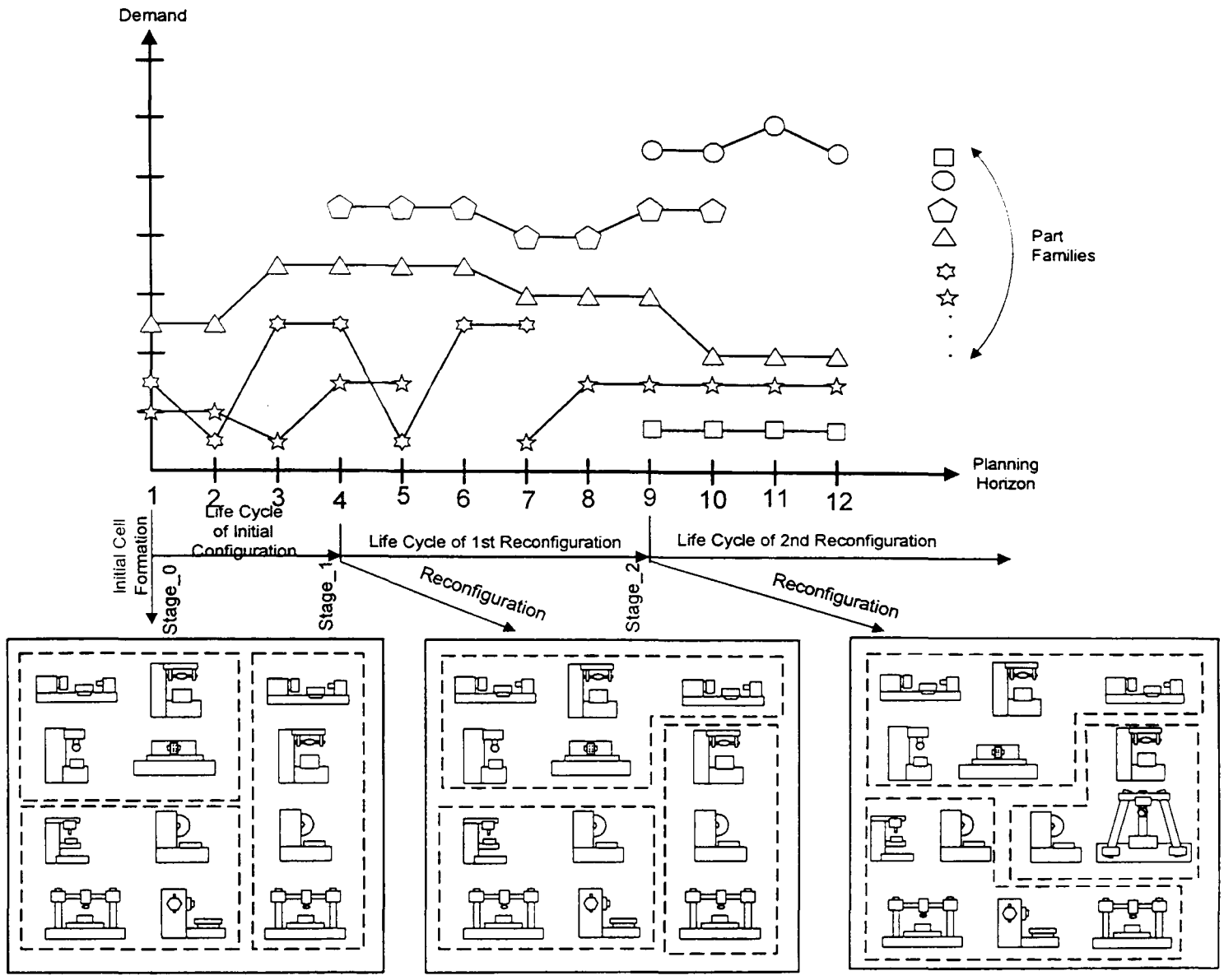


Figure 3.4 Formation of virtual cells in relation to changing production requirements

3.4 CONCLUSIONS

In this chapter, the multiple objective decision support framework for cell formation, cell loading and reconfiguration is introduced. Modules that made up the framework are briefly explained for motivational purposes. Each of these modules can also separately be used as stand-alone programs for their specific applications. Detailed explanations about their construction and application are given in the following chapters of this thesis.

CHAPTER FOUR

4. APPLICATION OF TABU SEARCH TO THE GENERAL PROBLEM OF MULTIPLE OBJECTIVE OPTIMISATION: DETERMINATION OF PARETO-OPTIMAL SET

4.1 INTRODUCTION

The design of complex engineering systems usually involves several competing objectives that cannot easily be combined into a single objective function. This section introduces the application of Tabu Search (TS) to solve the general problem of multiple objective optimisation (MOO) which finds the Pareto optimal set in MOO optimisation. In MOO applications there are usually many viable solutions for the problem. Pareto optimality is generally used to characterise these solutions. As explained in Chapter 2, Pareto optimal solutions generally consist of many solutions and additional techniques (like goal programming etc., refer to Chapter 2) are required to order them and determine a single solution which meets decision-maker's specifications. However, before attempting to use any special technique for finding a Pareto optimal solution to a MOO, its ability to determine the Pareto optimal set should be known. Having this in mind, this chapter investigates the applicability of TS in finding Pareto optimal solutions in MOO applications.

4.2 TABU SEARCH BASED APPROACH TO FIND PARETO OPTIMAL SET IN MOO

TS belongs to the class of problem-independent, heuristic, global-optimisation methods and can handle any type of objective functions and constraints (Glover, 1993, Bland and Davson, 1991, Siarry and Berthiau, 1997). It has generally been applied to combinatorial optimisation problems by considering a single objective function (Islam and Eskioglu, 1997, Reeves, 1995).

As explained in Chapter 2, the basic TS algorithm operates in the following way: it starts from a randomly selected, feasible seed solution (s); from this seed solution, a set of neighbourhood solutions (s^*) is generated using a number of previously determined movement strategies. The objective function is evaluated for each solution in s^* and the best neighbour replaces the seed solution, even though it may be worse than s: in this way it is possible to escape local minima (or maxima) of the objective function. The algorithm iterates until some given stopping condition is reached. There is a danger that the algorithm as described above may recycle old solutions and become trapped in a loop. To avoid this situation, the last m seed solutions are stored in a list, which is called the *tabu list*. The neighbours of the current solution that appear in the *tabu list* are systematically eliminated, so at each iteration the algorithm is forced to select a solution not recently explored. The main stages of the basic TS algorithm are: *initial solution*, *generation of neighbours*, *selection* and *updating*, as shown in Figure 4.1.

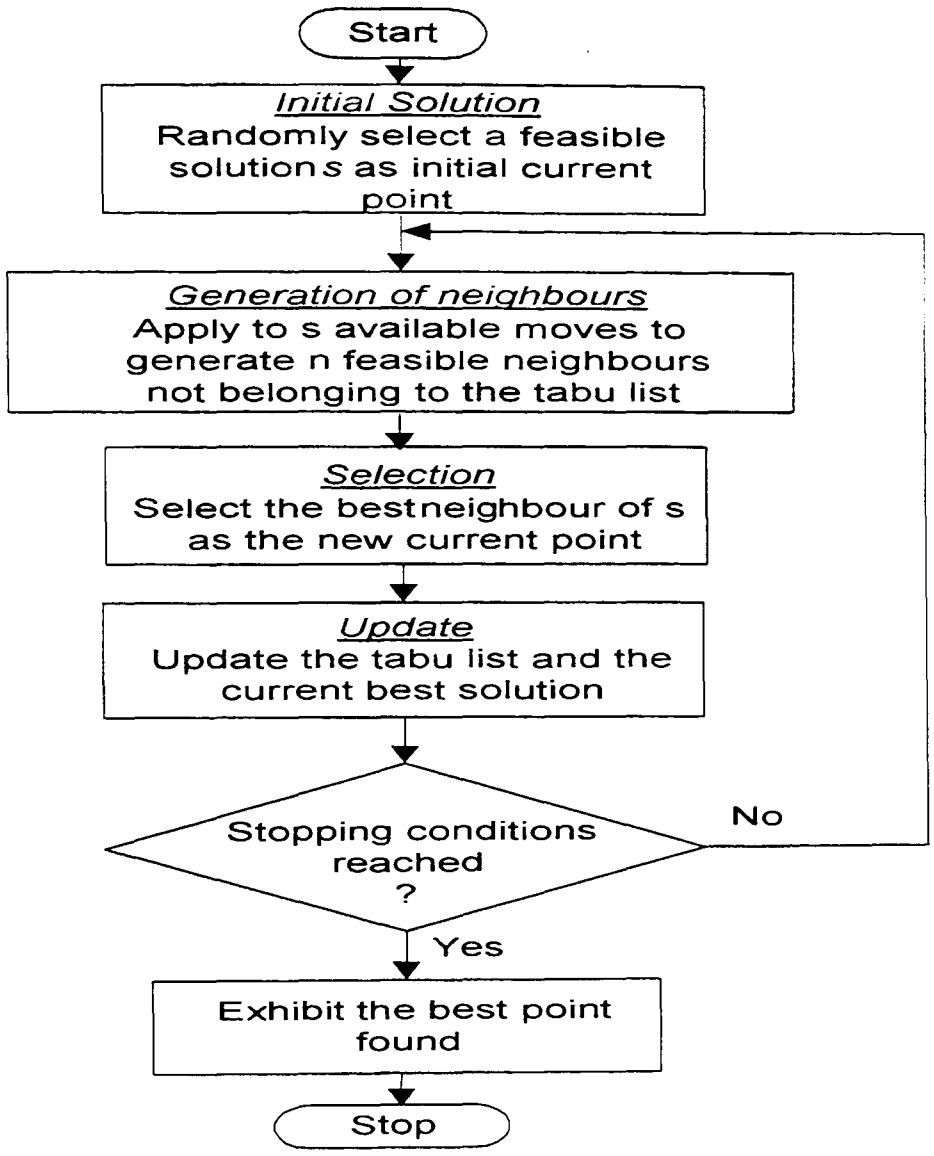


Figure 4.1 The flowchart of the basic TS algorithm

There are more complex versions of TS that improve its search capability, but in this chapter the focus is on the basic TS algorithm to demonstrate its application in finding Pareto optimal solutions for MOO problems. Most of the refinements can easily be incorporated into the proposed algorithm to handle specific application requirements.

4.2.1 Development of the TS Algorithm

"The idea of applying TS to multiple objective optimisation comes from its solution structure in working with more than one solution (neighbourhood solutions) at a time. This gives TS the opportunity to evaluate multiple objective functions simultaneously"

(Baykasoglu *et. al.*, 1999-a, b) To enable the TS algorithm to work with more than one objective, the *selection* and *updating* stages of the basic TS are redefined. The other stages are similar to the original algorithm. Two more lists are defined in addition to the *tabu list*. The first is the *Pareto list*, which collects selected non-dominated solutions found by the algorithm. The second is the *candidate list*, which collects all other non-dominated solutions that are not selected as Pareto optimal solutions in the current iteration. These solutions may become seed solutions if they maintain their non-dominated status in later iterations. The *candidate list* gives the opportunity to diversify the search.

The elements of the proposed TS algorithm for finding Pareto optimal solutions in MOO problems with different types of variables (integer, zero-one, discrete or continuous) and objective functions (linear, non-linear, convex, non-convex) are defined as follows:

Initial Solution:

Any randomly generated or known feasible solution vector.

Generation of neighbourhood solutions:

To create a neighbour for any type of variable, new values are formulated as follows:

- *Integer Variables:* A move is the change of a variable from one integer value to another, with a random step size.

$$x_i^* = x_i + \text{integer}[(2*random() - 1)*stepi_i] \qquad 4.1$$

- *Zero-One Variables:* A move is the change of a variable from zero to one, or vice versa.

$$x_i^* = \begin{cases} 1 & if\ x_i = 0 \\ 0 & if\ x_i = 1 \end{cases} \qquad 4.2$$

- *Discrete Variables:* A move is the change of a variable from one pre-specified discrete value to another.

$$x_i^* = d_{(l+\text{integer}[(2*random(\,)-1)*stepd_i])} \quad if \quad x_i = d_l \qquad 4.3$$

- *Continuous variables:* A move is the random change of a variable value.

$$x_i^* = x_i + (2 * random(\,) - 1) * stepc_i \qquad 4.4$$

where

x_i: Value of the i^{th} variable prior to the neighbourhood move.

x_i^*: Value of the i^{th} variable after the neighbourhood move.

$random(\,)$: Random number generator, where $random(\,) \in (0,1)$.

$stepi_i$: Integer step size of integer variable for the move.

$stepd_i$: Integer step size of discrete variable for the move.

$stepc_i$: Real value step size of continuous variable for the move.

d_l: The l^{th} element of the discrete variable subset X^d.

integer[ ]: Function to convert a real value to an integer value.

For multivariate problems, neighbourhood moves can be realised in two ways:

i- Changing an individual variable each time (orthogonal move)

ii- Changing all variables simultaneously (combined move)

For different types of problems (i.e. constrained, unconstrained), these two approaches may have different effects on the solution efficiency. Intuitively, for constrained problems, the combined move approach moves farther away from the current solution than does the orthogonal move. However, it may result in more infeasible neighbourhood moves, when the current solution is close to the boundaries of constraints or close to the global optimum (or the trade off curve). Thus it would require more computational time in each iteration when the algorithm is approaching the final solution or the trade off curve. Figure 4.2 shows the neighbourhood moves of two approaches with two variables respectively.

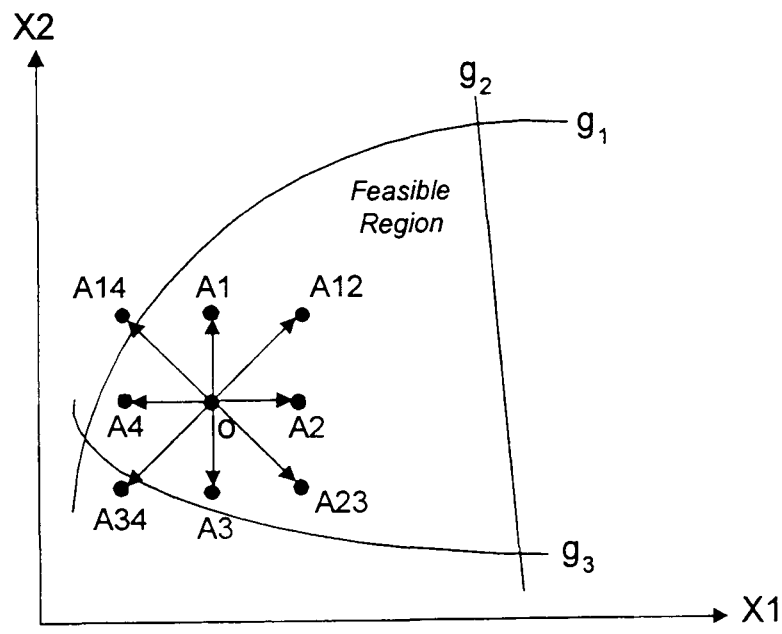


Figure 4.2 Neighbourhood move spaces of different move approaches

In Figure 4.2, O is the current solution. A2, A4 and A1, A3 indicate the points after orthogonal random moves (with the same step size) of variable X1 and X2, respectively. g_1, g_2, g_3 are the constraints which define the feasible region of solutions. A12, A14, A23, A34 are the points reached by neighbourhood moves changing X1 and X2 simultaneously. It can be seen from the Figure 4.4 that orthogonal moves A1, A3, A4 are within the constraint boundaries while combined

move, A14, A34 fall outside the feasible region. However, at an earlier stage of the search, the combined approach will move faster towards the optimum solution or trade off curve than will the first approach. This is because combined step size of the second approach is larger than the first one. In this study both approaches have been applied to the same problems. For constrained problems orthogonal move strategy requires less computational time to converge than the combined move approach and there is no significant difference between the solution qualities obtained by these two approaches. For unconstrained problems, combined move strategy leads the algorithm converging faster than the orthogonal approach. This research suggests that the orthogonal move approach should be employed for constrained problems and the combined move approach for unconstrained problems.

According to the types of variables used in the model, the appropriate movement strategies are used to generate a previously determined number of feasible, non-tabu, neighbourhood solutions from the current seed solution. Neighbourhood solutions must also be non-dominated by the current seed solution.

Selection of the seed solution:

Based on the Pareto optimality logic (domination and non-domination), the selection of the best neighbourhood solution as the new seed solution is performed in the following manner:

i. For each neighbourhood solution vector, the corresponding objective function values are calculated. In the example given below, the neighbourhood size is three and there are two real variables and two objective functions to be maximised.

Seed solution followed by objective function values		*Neighbourhood solutions (non-tabu, feasible and not dominated by the seed solution)*		*Objective function values of neighbourhood solutions*
(4.8 4.6)(52.4 40.93)	➠	(6.3 6.1)	➠	(60.08 47.09)
		(6 6)	➠	(58.79 46.54)
		(6.4 6)	➠	(60.39 46.86)

ii. Candidate seed solutions within the neighbourhood solutions are identified. Candidate seed solutions should not be dominated by other neighbourhood solutions, solutions in the Pareto list or solutions in the candidate list. This process is illustrated below, using the same example.

Seed solution followed by objective function values		*Neighbourhood solutions (non-tabu, feasible and not dominated by the seed solution)*		*Objective function values of neighbourhood solutions*
(4.8 4.6)(52.4 40.93)	➠	(6.3 6.1)	➠	(60.08 47.09) ❍
		(6 6)	➠	(58.79 46.54)
		(6.4 6)	➠	(60.39 46.86) ❍

(❍: Candidate solutions)

Pareto List	Candidate List
(0 0) (0 0)✖	(4 3) (46.93 33.98) ✖
(0.5 0.5) (16.97 13.44) ✖	(3 4) (42.64 36.93)
(1 1) (24 19) ✖	
(2 2) (33.94 26.87) ✖	
(3 3) (41.57 32.91) ✖	
(3.8 3.6) (46.57 36.26) ✖	
(4.8 4.6) (52.4 40.93)	

(✖ : Eliminated solution from previous iterations)

iii. One of the candidate solutions is randomly selected as the new seed solution. This process is illustrated below using the same example.

Seed solution followed by objective function values		*Neighbourhood solutions (non-tabu, feasible and not dominated by the seed solution)*		*Objective function values of neighbourhood solutions*
(4.8 4.6)(52.4 40.93)	➠	(6.3 6.1)	➠	(60.08 47.09) ❍
		(6 6)	➠	(58.79 46.54)
		(6.4 6)	➠	(60.39 46.86) ●

(●: Randomly selected new seed solution)

If there are no candidate solutions in the current neighbourhood, the oldest solution from the candidate list is selected as the seed solution.

It can be understood from the above selection strategy that the main interest is not dominated solutions, it is aimed to find the Pareto optimal solutions which do not dominate each other. Consequently, a single global optimum solution is not searched. The approach presented here is totally different from the single objective and many of the MOO approaches (refer to Chapter 2) which reduce the MOO problem to a single objective optimisation problem by applying some additional techniques (i.e. weighting, game theory etc.). In these approaches a non-improving solution (which is the best current neighbour in TS) can be accepted as the current solution if it is not tabu. If it is tabu and passes a previously defined aspiration criterion it is still accepted, even if it is worse than the previous current solution. This strategy works well and has the ability to find a global optimum solution in the single objective optimisation. But, it is not easy (or may not be possible) to evaluate multiple objectives simultaneously and determine the Pareto optimal solutions with this strategy. This is because it concentrates only on single objective function evaluations. For this reason the present strategy is offered which works with two more dynamic lists namely *Pareto list* and *Candidate list*. Their functioning is explained through this chapter. As is stated at the beginning of this section, the *Candidate list* (which collects potential candidate Pareto optimal solutions and updates their status dynamically) enables the search process to avoid abandoning potential solutions while searching and to diversify the search (this case is similar to attempting to avoid falling into trap of local optima in global optimisation). The

Pareto list collects the seed (or currently selected) potential Pareto optimal solutions and dynamically updates their status.

Updating the lists:

The initial feasible solution vector is recorded as the first known Pareto solution vector. In each iteration solutions that are dominated by any neighbourhood solution are removed from both the *Pareto* and *Candidate lists*. Then the seed solution is added to the *Pareto list*, and other candidate solutions are put into the *Candidate list*. This process is shown on the same example below.

Pareto List	Candidate List
(0 0) (0 0)✖	(4 3) (46.93 33.98) ✖
(0.5 0.5) (16.97 13.44) ✖	(3 4) (42.64 36.93) ✖
(1 1) (24 19) ✖	(6.3 6.1) (60.08 47.39)
(2 2) (33.94 26.87) ✖	
(3 3) (41.57 32.91) ✖	
(3.8 3.6) (46.57 36.26) ✖	
(4.8 4.6) (52.4 40.93) ✖	
(6.4 6) (60.39 46.86)	

(✖ : Eliminated solution)

Selected seed solutions for an arbitrarily defined number of previous moves are considered as tabu, since reusing one of them may trap the algorithm into cycling through recent moves. In the present algorithm, the tabu list contains m solutions, corresponding to the last m seed solutions. The tabu list is circular, i.e. when it is full a new item replaces the head of the list.

Termination:

If a previously determined number of iterations is reached, or if the candidate list is empty and the algorithm cannot find any new candidate solutions, the program terminates.

The flowchart of the developed algorithm is shown in Figure 4.3.

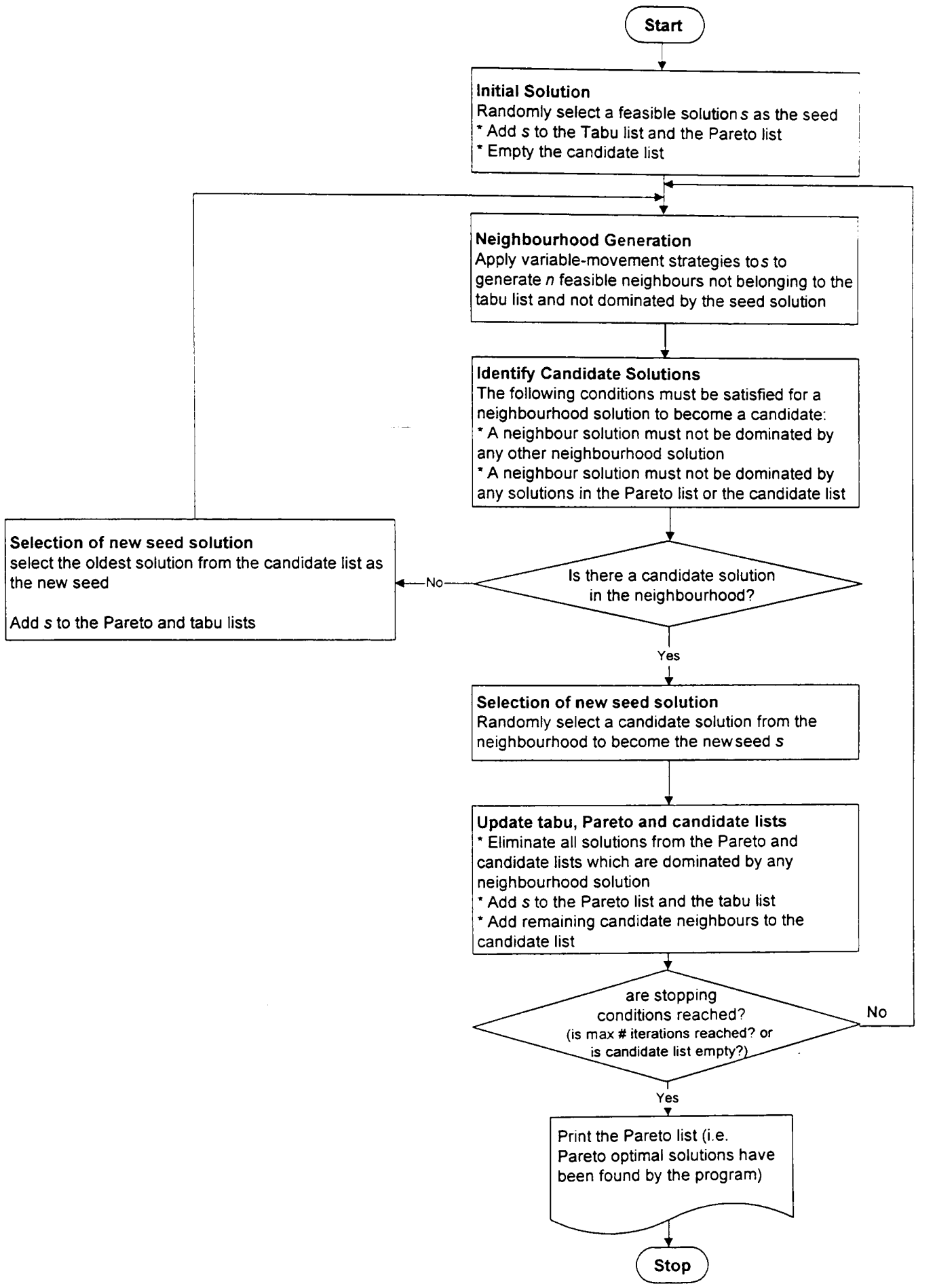


Figure 4.3 The flowchart of the TS algorithm to find Pareto optimal solutions

To illustrate how the proposed algorithm works, a small part of a step by step manual solution of the following test problem, given in Winston (1994), is depicted in Figure 4.4.

$$\max \quad f_1(x) = 20\sqrt{x_1} + 4\sqrt{x_2}$$

$$\max \quad f_2(x) = 4\sqrt{x_1} + 15\sqrt{x_2}$$

$$s.t.$$

$$100x_1 + 60x_2 \le 1000$$

$$x_1 \ge 0$$

$$x_2 \ge 0$$

$$x_1, x_2 \in \Re$$

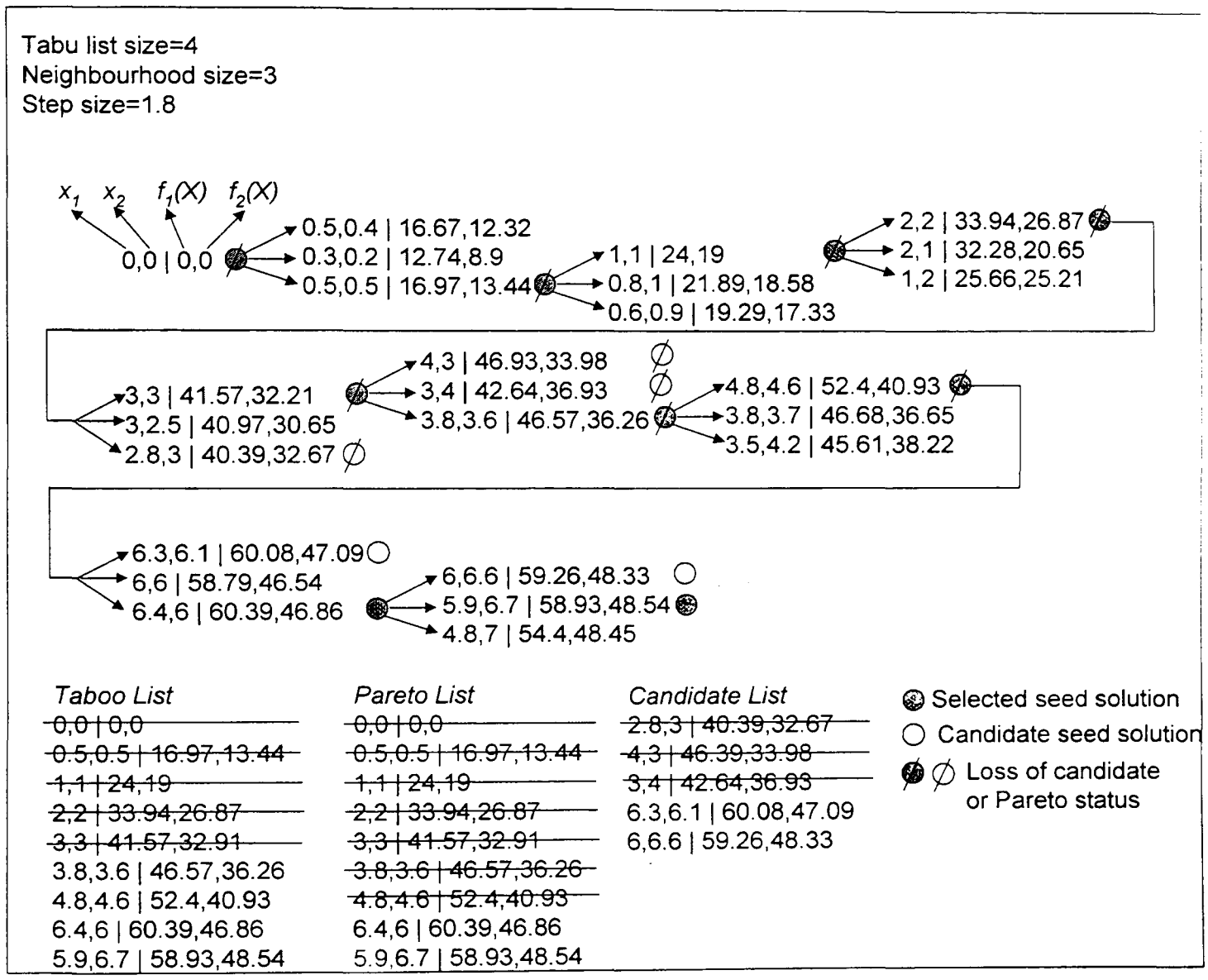


Figure 4.4 Part of a step by step manual solution for the example problem

The computer implementation of the developed algorithm for this example problem is depicted in Figure 4.5. All candidate solutions are shown as the algorithm converges towards the trade-off curve when the curve is reached, the candidate list provides the mechanism to explore along the Pareto front and generate a large set of Pareto optimal solutions. The shape of the trade-off curve is identical to the one

given in Winston (1994) using the GINO optimisation software and the constraint method, but the proposed algorithm found a much higher number of Pareto optimal solutions (i.e. 9 Pareto optimal solutions reported in Winston (1994), TS algorithm found 204 Pareto optimal solutions).

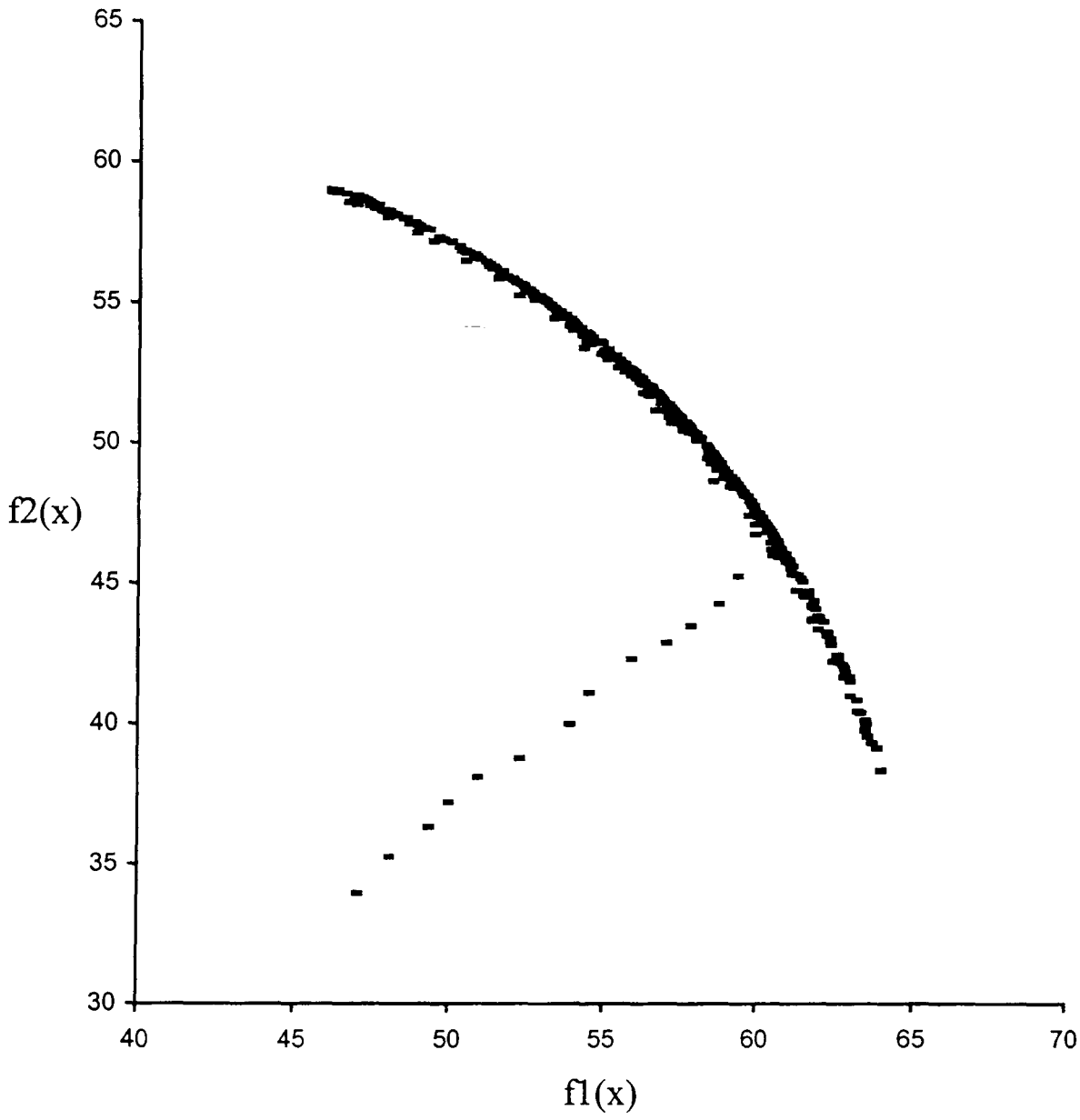


Figure 4.5 Computer simulation results for the test problem

To further demonstrate the behaviour of the algorithm while searching for the Pareto optimal solutions, a schematic representation of a problem with two maximisation objectives is depicted in Figure 4.6.

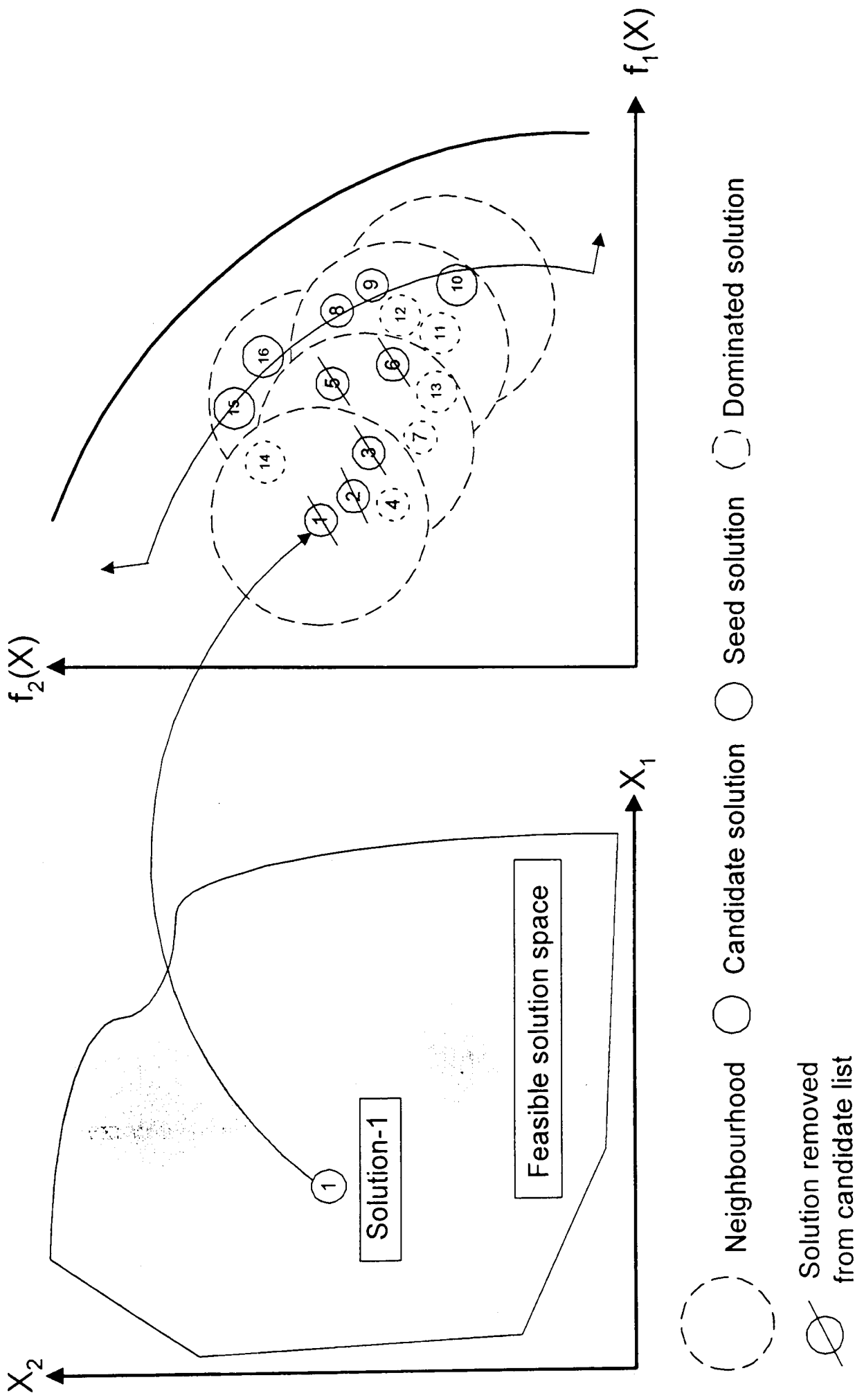


Figure 4.6 Behaviour of the algorithm while searching for Pareto optimal set in MOO

In Figure 4.6 the initial seed solution is shown by number 1. Solutions 2, 3 and 4 are generated from solution 1, all of which are feasible and not dominated by solution 1. Solution 4 is dominated by other neighbour solutions so it is not a candidate seed solution. There are two candidate seed solutions, namely solutions 2 and 3.

In the next iteration, solution 3 is selected randomly as the seed solution. Solutions 5, 6 and 7 are generated; solutions 5 and 6 are candidates. Solution 2 loses its candidate status and solution 3 loses its Pareto status (because both are dominated by solution 5). Solution 6 is randomly selected as the seed solution for the next iteration.

Solutions 8, 9 and 10 are generated from solution 6. Solutions 8, 9 and 10 are all candidate solutions (i.e. they are not dominated by the current seed, Pareto solutions or other candidate solutions) and solution 6 loses its Pareto solution status. Solution 10 is randomly selected as the current seed solution. Solutions 11, 12 and 13 are generated. There is no candidate seed solution in this iteration, so the oldest candidate solution (solution 5) is selected as the seed solution. Solutions 14, 15 and 16 are generated from solution 5. Solutions 1 and 5 lose their Pareto solution status. Solution 15 is selected as the current seed solution and so on. As can be seen from Figure 4.6, in every iteration solutions move towards the trade-off curve simultaneously.

4.2.2 Numerical Examples and Comparative Work

Three examples from the literature are presented below to show and compare the efficiency of the proposed algorithm. To effectively use the developed algorithm,

parameters such as variable step-size, number of solutions in the neighbourhood and the starting values of the variables should be carefully selected. It is possible to use trial runs of the algorithm as a guide to interactively select the most appropriate settings. A C++ computer program has also been developed for application to a wide range of problems during this study.

Example 1:

The following non-linear multiple objective model with two continuous variables has been previously solved by Murata, Ishibuchi and Tanaka (1996). They used genetic algorithms (MOGA, which was explained in Chapter 2) to find Pareto optimal solutions.

$$\begin{aligned} \min \quad & f_1(x) = 2\sqrt{x_1} \\ \min \quad & f_2(x) = x_1(1-x_2)+5 \\ & s.t. \\ & 1 \le x_1 \le 4 \\ & 1 \le x_2 \le 2 \\ & x_1, x_2 \in \Re \end{aligned}$$

The computer simulation results of the proposed algorithm for this test problem are depicted in Figure 4.7. The algorithm is converged 6 seconds. As can be seen from the graph, hundreds of solutions are obtained on the concave Pareto front. The shape of the trade-off curve is identical to the one given in Murata, Ishibuchi and Tanaka (1996), but the TS based algorithm found many more solutions (around 40% more) than the genetic algorithm (MOGA) method and the issue of weighting objectives is avoided. The Pareto optimal solution set lies generally on the boundary of the feasible solution space for a given MOO problem (Ignizo, 1982). In order to test the Pareto optimality of the obtained solutions the generated trade-off curve can be

inspected by trying to find some better possible solutions vectors around it. This simple test can give an idea about the Pareto optimality of the generated solutions. Two such inspections are presented here. In the first inspection the possibility of having a solution vector (x_1=?, x_2=?) which results better objective function values (i.e. ($f_1(x_1)$=3.4, $f_2(x_2)$=1.9)) is investigated. This objective function values are very close to the Pareto optimal solution ($f_1(x_{1p})$=3.5, $f_2(x_{2p})$=2) (x_{1p}=3.0625, x_{2p}=1.98 feasible) detected by the TS algorithm (see Figure 4.7). If we substitute these new objective function values and solve for the resulting equations (i.e. $3.4 = 2\sqrt{x_1}$ and $1.9 = x_1(1-x_2)+5$) for x_1 and x_2 the result will be (x_1 = 2.89, x_2= 2.073) this is an infeasible solution (i.e. second constraint is violated, x_2 is greater than 2), therefore it is possible to say that the TS algorithm has detected a potential Pareto optimal solution. In the second inspection the possibility of having a solution vector (x_3=?, x_4=?) which results objective function values ($f_1(x_3)$=2.4, $f_2(x_4)$=3.4) which are better then the ones ($f_1(x_{3p})$=2.5, $f_2(x_{4p})$=3.5) (x_{3p}=1.5625, x_{4p}=1.96 feasible) detected by the TS algorithm is investigated. The following solution is obtained (x_3=1.44, x_4=2.111) this is again an infeasible solution (i.e. x_4 is greater than 2), therefore it is possible to say that the TS algorithm has detected another Pareto optimal solution. Several other inspections were also made but in all trials no better solutions obtained. Although this tests give an idea about the Pareto-optimality of the generated solutions further research is necessary to guarantee the Pareto-optimality of the generated solutions.

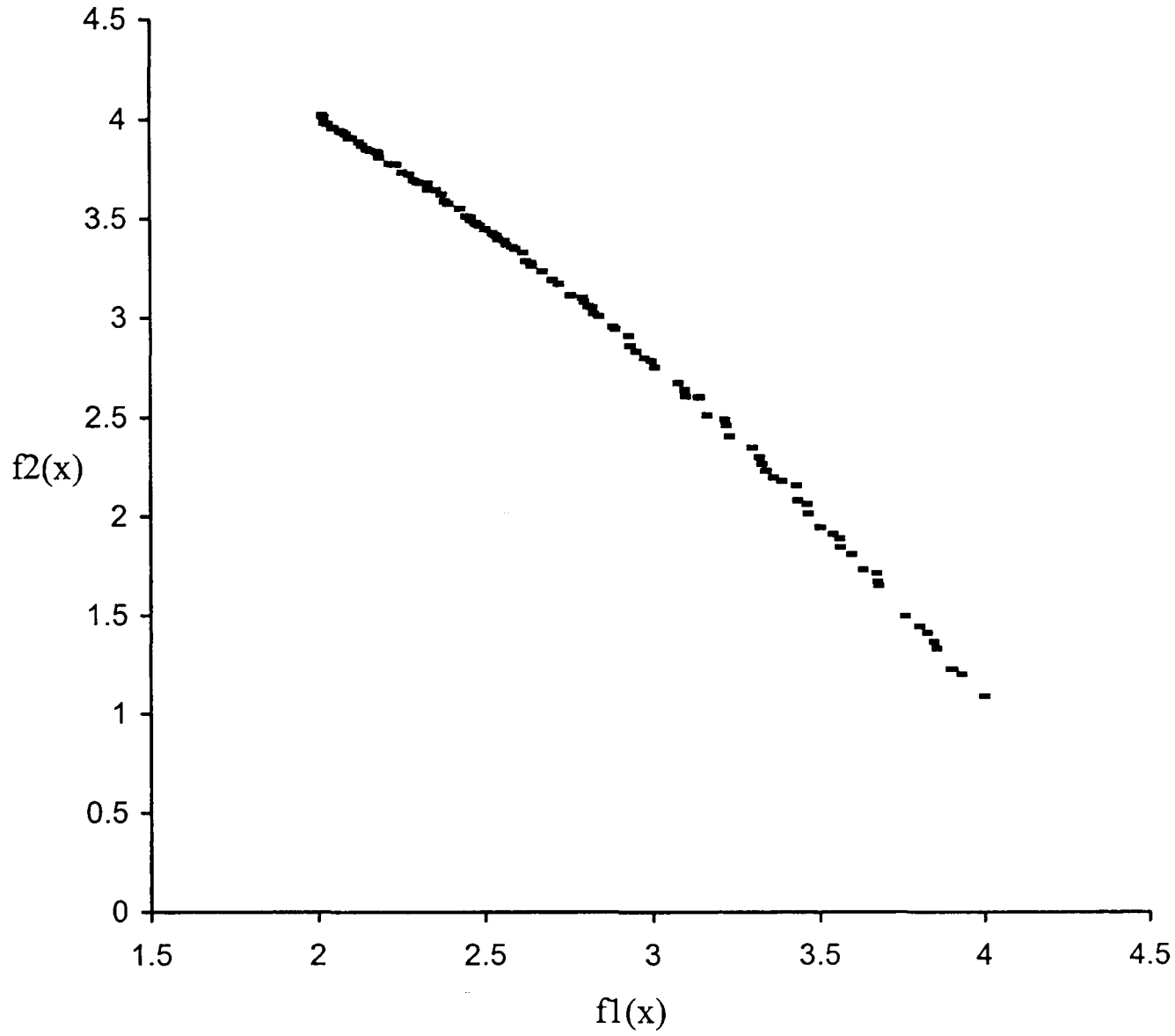


Figure 4.7 Computer simulation result for Example 1

Example 2:

The following model is a non-linear problem with six continuous variables that was given by Osyczka and Kundu (1996).

$$\min \quad f_1(x) = -25((x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2)$$
$$\min \quad f_2(x) = (x_1-1)^2 + (x_2-1)^2 + (x_3-1)^2 + (x_4-1)^2 + (x_5-1)^2 + (x_6-1)^2$$
$$s.t.$$
$$x_1 + x_2 - 2 \geq 0$$
$$6 - x_1 - x_2 \geq 0$$
$$2 - x_2 + x_1 \geq 0$$
$$2 - x_1 + 3x_2 \geq 0$$
$$4 - (x_3 - 3)^2 - x_4 \geq 0$$
$$(x_5 - 3)^2 + x_6 - 4 \geq 0$$
$$0 \leq x_1 \leq 10$$
$$0 \leq x_2 \leq 10$$
$$0 \leq x_3 \leq 10$$
$$0 \leq x_4 \leq 10$$
$$0 \leq x_5 \leq 10$$
$$0 \leq x_6 \leq 10$$
$$x_1, \ldots\ldots, x_6 \in \Re$$

The simulation results of the TS algorithm for this test problem are shown in Figure 4.8. As can be seen from the figure, the trade-off curve is detected successfully, with hundreds of Pareto optimal solutions. Osyczka and Kundu (1996) applied genetic algorithms to determine the Pareto optimal set for this problem. The maximum number of Pareto optimal solutions they found was only 34. The proposed TS algorithm is converged in 35 seconds for this problem.

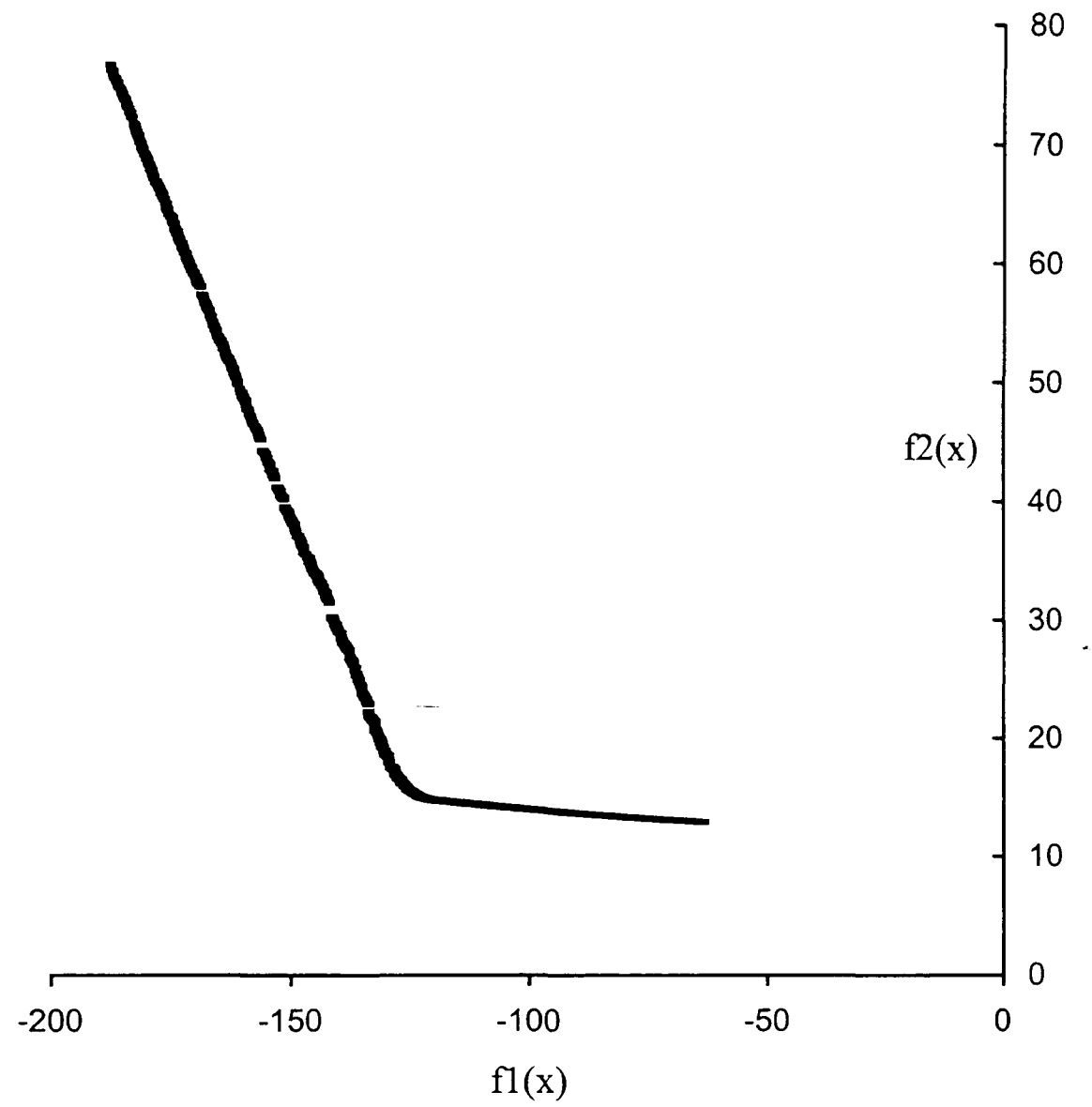


Figure 4.8 Computer simulation result for Example 2

Example 3:

This test is a design optimisation problem for a multiple disc brake that was previously solved by Osyczka and Kundu (1996). They used both a plain stochastic method and genetic algorithms. The problem uses a mixture of continuous and integer variables; details of the problem can be found in Osyczka and Kundu (1996). The model is given as follows:

$$\min \quad f_1(x) = 4.9*10^{-5}(x_2^2 - x_1^2)(x_4 - 1)$$
$$\min \quad f_2(x) = (9.82*10^6(x_2^2 - x_1^2)) / (x_3 x_4 (x_2^3 - x_1^3))$$
$$\min \quad f_3(x) = x_3$$
$$s.t.$$
$$(x_2 - x_1) - 20 \geq 0$$
$$30 - 2.5(x_4 + 1) \geq 0$$
$$0.4 - x_3 / 3.14(x_2^2 - x_1^2) \geq 0$$
$$1 - 2.22*10^{-3} x_3 (x_2^3 - x_1^3) / (x_2^2 - x_1^2)^2 \geq 0$$
$$2.66*10^{-2} x_3 x_4 (x_2^3 - x_1^3) / (x_2^2 - x_1^2) - 900 \geq 0$$
$$55 \leq x_1 \leq 80$$
$$75 \leq x_2 \leq 110$$
$$1000 \leq x_3 \leq 3000$$
$$2 \leq x_4 \leq 20$$
$$x_1, \ldots, x_3 \in \Re$$
$$x_4 \quad \text{integer}$$

For the above problem Osyczka and Kundu (1996) reported finding 19 Pareto optimal solutions using the plain stochastic method, and 133 solutions with the genetic algorithms method after 20000 evaluations. The tabu search based algorithm with the following parameters: neighbourhood size=20, tabu list size=20, step size for real variables 0.01, step size for integer variable=1, found 5964 Pareto optimal solutions after 20000 evaluations which took 82 seconds. Osyczka and Kundu also reported the extreme points (i.e. minimum value points for each separate criterion) obtained from both methods. The extreme points obtained by the three methods are compared in Table 4.1, and the comparison is also depicted in Figure 4.9.

Table 4.1 Comparison of extreme points obtained from plain stochastic method, genetic algorithms and multiple objective TS

Method	Minima	$X[x_1,x_2,x_3,x_4]^T$	$F(X)=[f_1(x),f_2(x),f_3(x)]^T$
Plain stochastic method (PSM)	min $f_1(x)$	[62.6,83.5,2920.9,11]	[**1.79**,2.77,2920.9]
	min $f_2(x)$	[70.4,106.6,2948.4,11]	[3.76,**2.24**,2948.4]
	min $f_3(x)$	[75.9,106.3,2309.2,11]	[3.25,2.80,**2309.2**]
Genetic algorithms (GA)	min $f_1(x)$	[65.8,86.1,2982.4,10]	[**1.66**,2.87,2982.4]
	min $f_2(x)$	[78.7,108.3,2988.3,11]	[3.25,**2.11**,2988.3]
	min $f_3(x)$	[72.6,109.2,2255.1,11]	[3.91,2.86,**2255.1**]
Multiple objective TS (MOTS)	min $f_1(x)$	[56.3042,76.4646,1183.29,2]	[**0.131156**,41.3532,1183.29]
	min $f_2(x)$	[79.9156,103.962,2981.64,11]	[2.16656,**2.15876**,2981.64]
	min $f_3(x)$	[63.6256,86.0813,1000.03,8]	[1.15309,10.8508,**1000.03**]

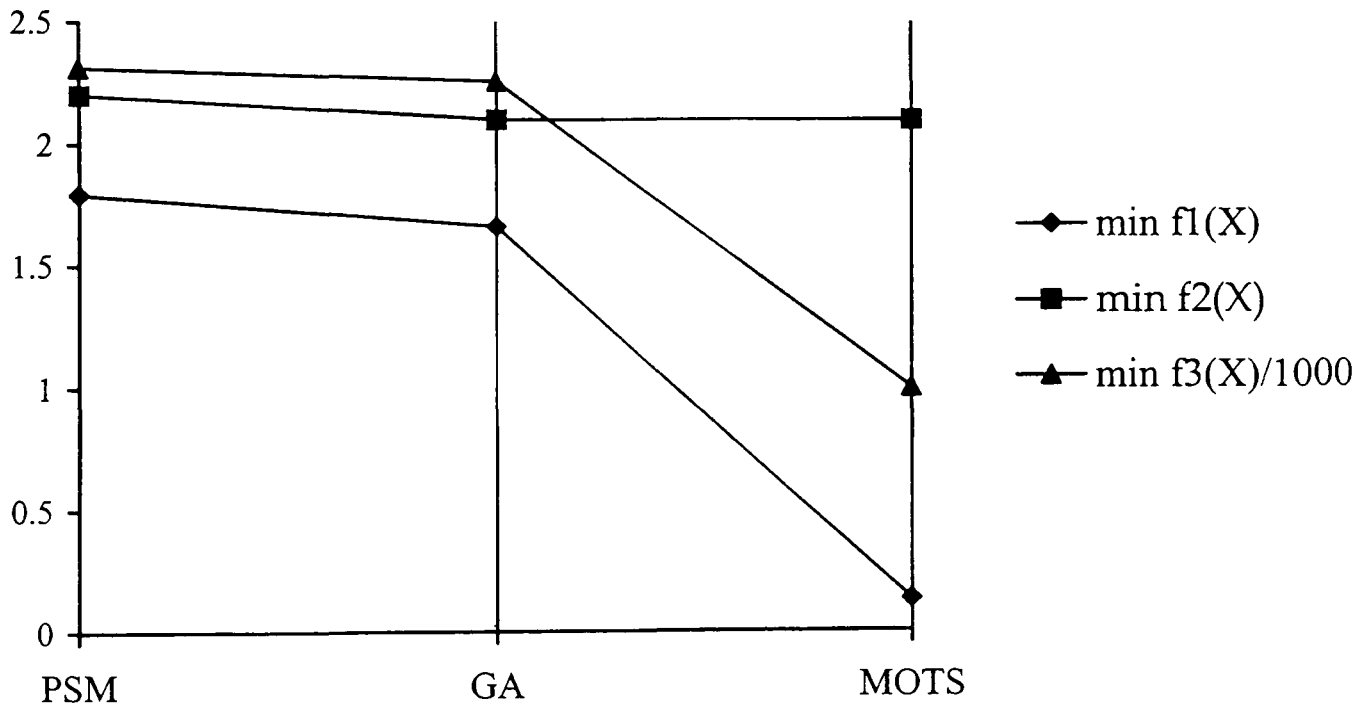


Figure 4.9 Graphical comparison of extreme points obtained from PSM, GA and MOTS

The trade-off surface obtained from computer simulations for this test problem is given in Figure 4.10.

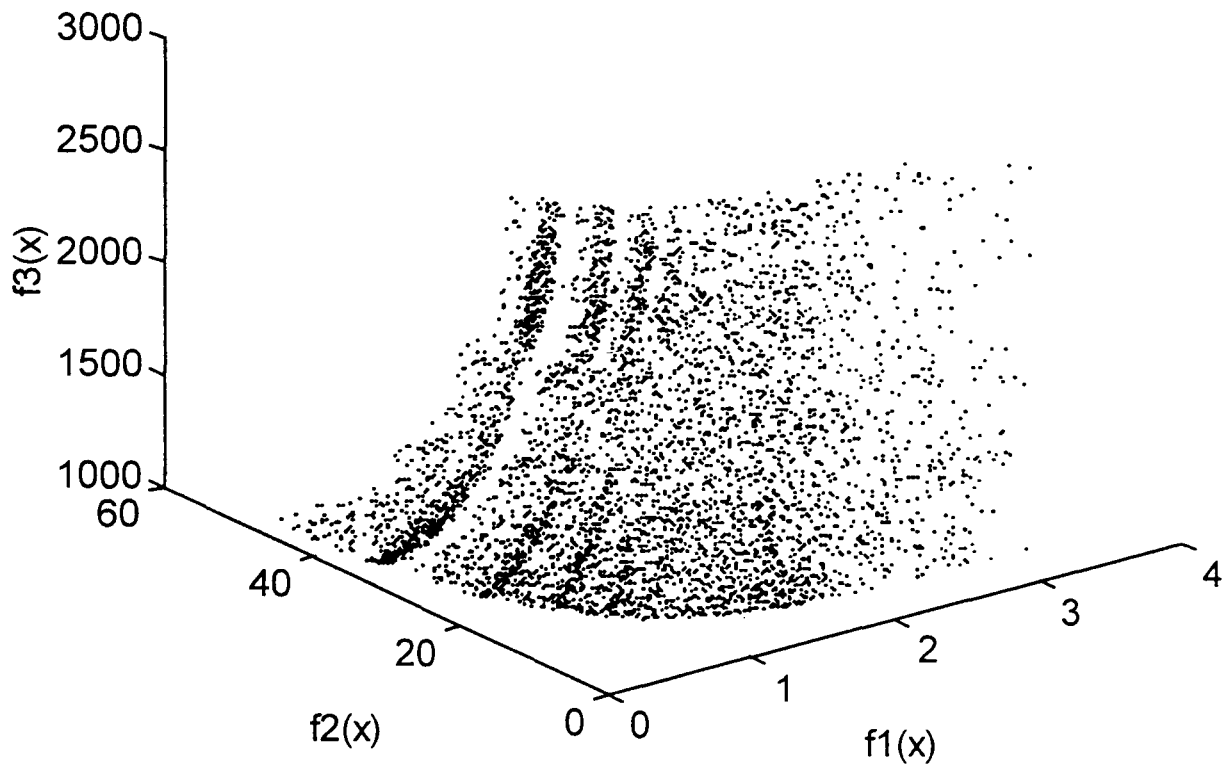


Figure 4.10 Computer simulation result for Example 3

4.3 CONCLUSIONS

Any optimisation technique that is able to work with more than one solution vector in its inherent solution mechanism, like TS, can be used for solving MOO problems. Based on this observation the applicability of TS to the general problem of MOO (i.e. finding a Pareto optimal set) is investigated in this chapter. This study is also the first direct application of TS to MOO without requiring additional techniques like weighting, game theory etc. It is shown that TS has a big potential to solve MOO problems.

The proposed TS algorithm to find a Pareto optimal set is explained in detail in this chapter. A comparative study has also been carried out. In almost every application the solutions found were at least as good, if not better than, the reported results. In

some test problems more than %50 improvement in solution quality is obtained. The TS based algorithm also finds far more Pareto optimal solutions than the techniques that are compared in this chapter. The computational time requirements of the proposed algorithm seems also reasonable it takes a couple of seconds to solve a small problem and a couple of minutes for a moderate size problem. However a detailed study for the computational time estimation is necessary. The proposed algorithm is implemented in the C/C++ computer programming language and all the computational work is performed on a Pentium-200 PC with 32MB Ram.

CHAPTER FIVE

5. DEVELOPMENT OF THE TABU SEARCH ALGORITHM TO SOLVE PRE-EMPTIVE GOAL PROGRAMMING MODELS

5.1 INTRODUCTION

A Pareto optimal set generally consists of many viable solutions for a Multiple Objective Optimisation (MOO) problem. Therefore an additional technique is required to order the Pareto optimal set and determine a single solution that closely meets the decision-maker's objectives. In Chapter 4 it has been shown that TS has the ability to solve MOO. In this chapter TS is applied to pre-emptive goal programming models (PGP). PGP is one of the best-known techniques to model MOO problems (Ignizo, 1982). Its ability to find a Pareto optimal solution in MOO applications has been proven (Ignizo, 1976). It tries to find a Pareto optimal solution that satisfies multiple objectives which are ordered based on the preference of the decision-maker in a multiple objective decision making problem (McMillan, 1975).

TS is a heuristic, general-purpose optimisation technique which works with a neighbourhood of solutions to optimise a given objective function. It has been widely applied to single objective optimisation problems in the literature. It has been shown in Chapter 4 that TS can also be directly applied to MOO problems. Because it handles more than one solution at a time, this gives it an opportunity to evaluate

multiple objective functions simultaneously. In this chapter, the development of the TS algorithm to solve pre-emptive goal programming models is shown. Application of the proposed TS algorithm to analytic and simulation optimisation problems is explained and some example applications are presented in order to test its performance. This algorithm (with some problem-related enhancements) is employed to solve the manufacturing cell formation, loading and reconfiguration problems studied in this thesis.

5.2 A TABU SEARCH BASED APPROACH TO SOLVE PREEMPTIVE GOAL PROGRAMMING MODELS

Goal programming (GP) was proposed by Charnes and Cooper (1961) to solve MOO problems. It has been studied by many researchers (Ijiri, 1965, Ignizo, 1976,1982) and successfully applied to many diverse, real-life problems (Myint and Taboocanon, 1994, Gokcen and Erel, 1997, Kornbluth, 1982, Ramanathan and Ganesh, 1995). Pre-emptive goal programming is a special case of GP, in which goals are ordered based on the preference of the decision-maker and they are optimised simultaneously favouring higher order goals. In non pre-emptive models (Archimedian goal programming), the goals are assigned weights and summed up then considered simultaneously as explained in Chapter 2. In this study, focus is on pre-emptive goal programming, due to the difficulties associated with the determination of weights in non pre-emptive goal programming. However, the proposed algorithm can also be applied to Archimedian goal programming models.

The most widely known methods for solving GP models are as follows:

- Simplex based approaches (separable programming, approximation programming etc. (Ijiri, 1965)
- Direct search approach (modified pattern search (Clayton, Weber and Taylor, 1982))
- Interactive approach (Dyer, 1972)
- Gradient based approach (Ijiri, 1965, Ignizo, 1982)

These methods have two main limitations. In many cases, they cannot guarantee to find the global or near-global optimum solution because they are local optimisation methods (Homaifar *et. al.,* 1994). Many of them are dependent on the form of the mathematical model (linear, integer, non-linear, etc.), so they require the analytical form of the model, otherwise the solution is not possible (Michalewicz, 1996). In many real-life problems the analytical form of some objectives and constraints may not be available, in such cases, simulation is employed to determine the values of these functions. An effective solution technique should be able to operate in such cases and should not be dependent on the form of the analytical model, as discussed in Chapter 2. The proposed TS-based technique overcomes these disadvantages of traditional techniques and can be applied to a variety of pre-emptive goal programming models. It can also be used in simulation optimisation applications as explained in a separate subsection later in this chapter.

5.2.1 Pre-emptive Goal Programming (PGP)

PGP is a powerful mathematical programming method developed to solve problems with conflicting linear or non-linear objectives and linear or non-linear constraints. The user is able to provide levels, or targets, of achievement for each objective and can prioritise the order in which goals are to be achieved. It tries to find an optimal

solution that satisfies as many of the goals as possible, by favouring the order specified. It is formally given as follows:

$$\text{lexmin}\{z_1 = \sum_{i=1}^{m}(w_{1i}^{+}d_i^{+} + w_{1i}^{-}d_i^{-}), z_2 = \sum_{i=1}^{m}(w_{2i}^{+}d_i^{+} + w_{2i}^{-}d_i^{-}), \ldots\ldots, z_q = \sum_{i=1}^{m}(w_{qi}^{-}d_i^{+} + w_{qi}^{-}d_i^{-})\}$$

such that;

$$\begin{aligned} & f_i(X) + d_i^{-} - d_i^{+} = b_i && i = 1,2,\ldots\ldots,m \\ & g_j(X) \le 0 && j = 1,2,\ldots\ldots,m_1 \\ & h_k(X) = 0 && k = 1,2,\ldots\ldots,m_2 \\ & d_i^{-}, d_i^{+} \ge 0 && i = 1,2,\ldots\ldots,m \end{aligned} \qquad 5.1$$

where

$z_1, z_2, .., z_q$ are in the priority order that they will be optimised ($z_1 >>> z_2 >>> ....$ $.. >>> z_q$)

d_i^{+} is the positive deviation variable representing the over-achievements of goal i

d_i^{-} is the negative deviation variable representing the under-achievements of goal i

w_{qi}^{+} is the positive weight assigned to d_i^{+} at priority q

w_{qi}^{-} is the negative weight assigned to d_i^{-} at priority q

X is an n-dimensional decision vector

f_i is a function of goal constraints

g_i is a function of real inequality constraints

h_i is a function of real equality constraints

b_i is the target value of goal i

m is the number of goal constraints

m_1 is the number of real inequality constraints

m_2 is the number of real equality constraints

5.2.2 The TS Algorithm

To enable the standard TS algorithm to work with more than one goal (objective) and to solve PGP models, *selection* and *updating* stages of traditional TS algorithm (refer to Figure 4.1 of Chapter 4) are redefined in this study. Other stages are similar.

The elements of the proposed TS algorithm for solving any kind of pre-emptive goal programming problems (linear/integer/non-linear/zero-one) are defined as follows:

Initial Solution:

Any randomly generated feasible solution vector or, a previously known good solution vector. Knowledge about a good initial solution vector can decrease computational time and may increase the speed of convergence (Reeves, 1995).

Generation of neighbourhood solutions:

Based on the types of variables used in the model, a previously determined number of feasible, non-tabu, neighbourhood solutions (*nneigh*) are generated from the current solution by applying the movement strategies that were explained in Chapter 4, section 4.2.1.

Selection of the current best solution vector:

Based on the pre-emptive goal programming logic, the selection of the best current solution vector from the neighbourhood solutions is performed in the following manner.

i. For each neighbourhood solution vector, calculate the goal deviations in the order of priorities specified in the problem. In the example given below, neighbourhood size is four and there are four integer variables and five goals.

Neighbourhood solutions (non Tabu & feasible)		Goal deviations (in the priority order)
(5 8 9 11)	➠	(3 2 0 1 2)
(5 9 9 11)	➠	(4 3 0 4 2)
(6 9 9 11)	➠	(2 3 2 2 1)
(6 9 9 12)	➠	(2 3 0 3 1)

ii. Check the first priority goal deviation for each neighbour's solution vector, and select the one that results in the minimum deviation. If there is more than one alternative neighbour solution for the first priority goal, check the second priority goal and choose the one with the smallest deviation, and so on. This process is illustrated below using the same example.

Neighbourhood solutions (non Tabu & feasible)		Goal deviations (in the priority order)
(5 8 9 11)	➠	(3 2 0 1 2)
(5 9 9 11)	➠	(4 3 0 4 2)
(6 9 9 11)	➠	(2 3 2 2 1)
*(6 9 9 12)	➠	(2 3 0 3 1)

↘ Replaces the current solution vector

Updating the best known solution vector:

The initial feasible solution vector is also recorded as the best-known initial solution vector. By applying the methodology described above, the best-known solution vector is updated in each iteration provided it gives a more optimal solution. This process is shown on the same example below.

1st case:

The best known solution vector and its goal deviations (in priority order)	The current best solution vector and its goal deviations (in priority order)
(6 9 9 12) ➠ (2 3 0 3 1)	(6 9 9 14) ➠ (2 3 0 5 1)

The best known solution is not improved (do not update)

2nd case:

The best known solution vector and its goal deviations (in priority order)	The current best solution vector and its goal deviations (in priority order)
(6 9 9 12) ➠ (2 3 0 3 1)	(6 9 9 14) ➠ (1 3 0 0 1)

↘ The best known solution

The best known solution is improved, so update the solution.

Tabu list:

Accepted solutions for an arbitrarily defined number of previous moves are considered as Tabu, because to allow one of them may trap the algorithm into cycling through recent, previous moves. In our algorithm, the Tabu list contains *m* solutions, corresponding to the *m* last 'current best' solutions: the Tabu list is circular, when it is full a new item replaces the head of the list.

Termination:

If a previously determined number of iterations (*iter*) is reached, or if there is no improvement in the best known solution in the last *t* iterations the algorithm terminates.

Guidelines for the determination of tabu search parameters:

Tabu list size (*m*), number of iterations (*iter*), convergence criteria (*t*), and neighbourhood size (*nneigh*) and step size for the variables constitute tabu search parameters. Determination of these parameters generally depends on the problem at hand. There is not a known unique analytic strategy or methodology to fix these parameters in the available literature. It is a common practice to solve the problem with different sets of these parameters to find the best possible combination of the parameter set. Within these parameters, determination of step sizes and neighbourhood size for real and integer variables is particularly important. If the ranges of variables are too wide and neighbourhood size is small and corresponding step sizes are chosen very small then computational time may increase considerably to find a good solution. With the same conditions, if step sizes are chosen too big then it may be possible to miss the optimal solution. As guidance for setting up the Tabu search parameters the usage of the following rule is advised after this study.

- **RULE:** If the ranges of variables are wide then using big step sizes is suggested, otherwise smaller step sizes can be preferred. If step sizes are big then the number of neighbourhood solutions in each iteration should be increased otherwise smaller number of neighbourhood solutions may safely be used. Make sure that the convergence criteria are satisfied before terminating, therefore the number of iterations should be big enough to assure convergence.

To illustrate how the algorithm works, a step by step manual solution is performed on the following integer pre-emptive goal programming problem, given in Winston's book (1994);

$$\text{lexmin}\{z_1 = (d_1^- + d_1^+), z_2 = (d_2^- + d_2^+), z_3 = (d_3^- + d_3^+)\}$$

$$s.t.$$

$$7x_1 + 3x_2 + d_1^- - d_1^+ = 40$$

$$10x_1 + 5x_2 + d_2^- - d_2^+ = 60$$

$$5x_1 + 4x_2 + d_3^- - d_3^+ = 35$$

$$100x_1 + 60x_2 \leq 600$$

$$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+ \geq 0$$

The optimum solution of the problem is given as; x_1=6, x_2=0, z_1=2, z_2=0, z_3=5

The solution steps of the proposed Tabu search algorithm are depicted in Figure 5.1. As can be seen from Figure 5.1, the Tabu search algorithm can easily converge to the optimum solution.

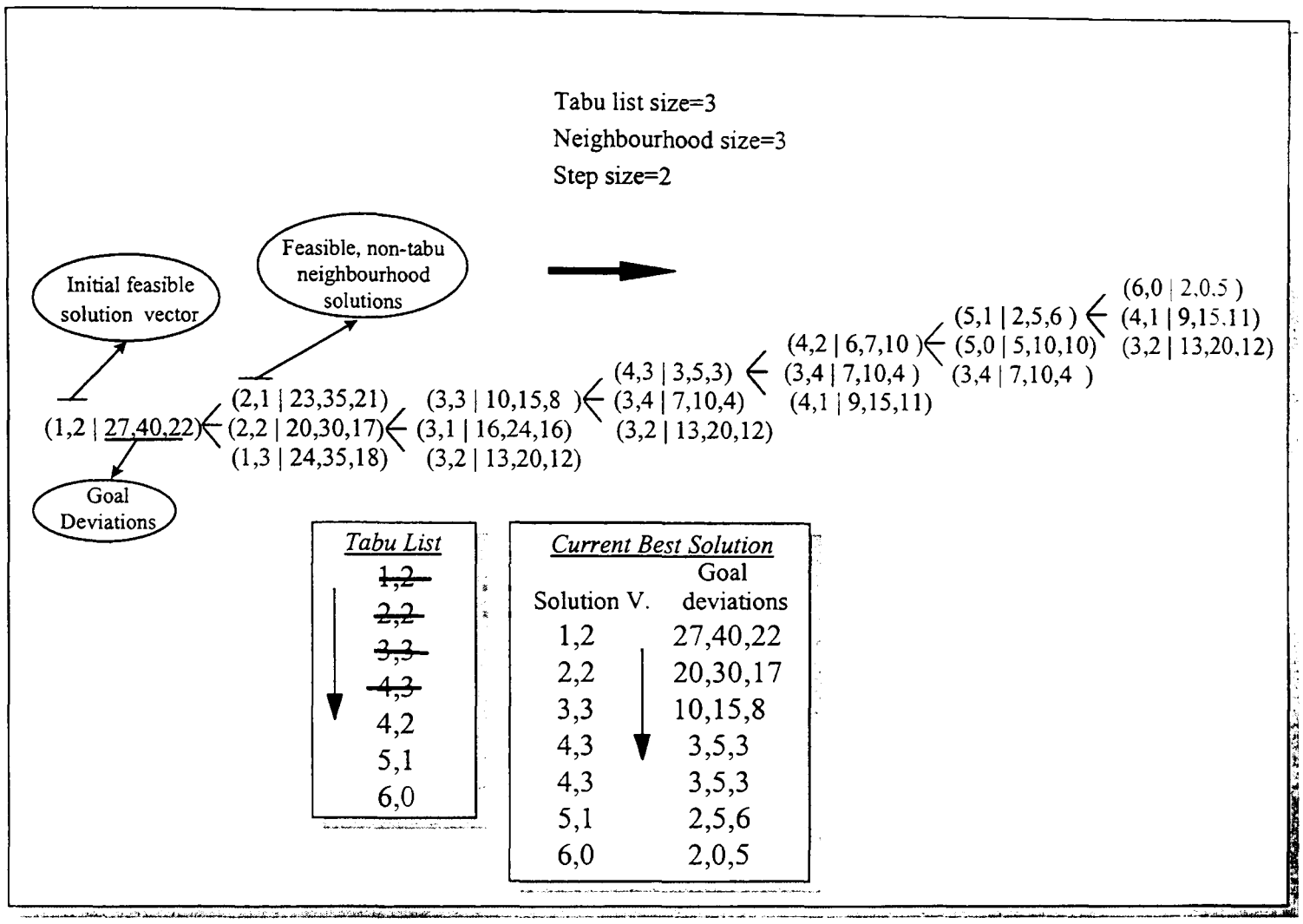


Figure 5.1 Step by Step manual solution of the example problem

5.2.3 Numerical examples and comparative work

The proposed algorithm has been applied to various types of pre-emptive goal programming problems collected from the literature. In each application the algorithm successfully found good solutions in comparison to the reported solutions. By using the C/C++ computer programming language a general-purpose computer program has also been developed for solving various types of pre-emptive goal programming problems. The computer program is implemented on a Pentium PC P5-200 with 32 MB RAM. Three test problems with continuous, integer and zero-one variables are presented below. Additional test problems are given in Appendix II.

Example 1

The following model is given in El-Sayed *et. al.*(1989)'s paper for optimising a three-bar truss. They used the linear goal programming techniques, with successive linearisation for the non-linear equations, to obtain the solution for the non-linear goal programming problem.

$$\text{lexmin}\{z_1 = (d_1^- + d_2^- + d_2^+ + d_3^- + d_3^+ + d_4^- + d_4^+)\}$$
$$s.t.$$
$$5.28x_1^2 + 3.74x_2^2 + 5.28x_3^2 + d_1^- - d_1^+ = 3.575$$
$$(0.178 / x_1^2) + d_2^- - d_2^+ = 1.0$$
$$(0.255 / x_2^2 + d_3^- - d_3^+ = 1.0$$
$$(0.178 / x_3^2) + d_4^- - d_4^+ = 1.0$$
$$x_1, x_2, x_3, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+ \geq 0$$

A detailed solution summary is given in Table 5.1. Upper half of the table presents the known solution. In the lower part of the table TS solution is reported. In order to reflect the effect of different parameter settings on the solution quality four different parameter settings are reported. Only the parameters that have the most significant effect on the solution are reported here (i.e. step size (for real and integer variables) and number of neighbourhood solutions in each iteration). Within these parameters '*step size*' is the most important one for continuous and integer variables. Number of iterations is taken big enough to assure convergence (i.e. 8000) in all applications. Convergence criterion is set to 100 iterations (i.e. if there is no improvement in the best solution vector in 100 consecutive iterations then algorithm terminates.). Tabu list size is set to 10. The same setting is used in all test problems.

Table 5.1 Solution summary for Example 1

Reported solution	**Goal satisfaction Levels**	**Computation time**	**Explanations**
x_1=0.4239, x_2=0.5047,x_3=0.4239	z_1=0.725	Not reported	
Tabu search solutions			
Parameter set: *Stepc*=0.3, *nneigh*=10 Solution: x_1=0.482522, x_2=0.615669, x_3=0.418525	z_1=0.582107	4 sec.	19.72% improvement
Parameter set: *Stepc*=0.3, *nneigh*=5 Solution: x_1=0.68565, x_2=0.508105, x_3=0.42306	z_1=0.639129	3.5 sec.	11.84% improvement
Parameter set: *Stepc*=0.1, *nneigh*=7 Solution: x_1=0.430716, x_2=0.507644, x_3=0.561096	z_1=0.485615	3.7 sec.	33.02% improvement
Parameter set: *Stepc*=0.1, *nneigh*=4 Solution: x_1=0.544277, x_2=0.51736, x_3=0.43777	z_1=0.517619	3.2 sec.	28.6% improvement

As the results indicate setting of Tabu search parameters affects the solution quality with continuous variables. In this problem, the range of variables is small (i.e. between 0 and 1), therefore selection of a smaller step size gives better results (as stated by the parameter setting rule in the previous section). It is also better to increase the neighbourhood size. However, good solutions can still be obtained with a relatively smaller neighbourhood size. It is also worth mentioning that at all settings Tabu search performed consistently better.

Example 2

The following model is given in Daellenbach *et. al.* (1983)'s book for solving an advertising media selection problem. They used the simplex method for solving this linear integer pre-emptive goal programming problem.

$$\text{lexmin}\{z_1 = (d_1^- + d_1^+), z_2 = (d_2^- + d_2^+), z_3 = (2d_3^- + d_3^+)\}$$
$$s.t.$$
$$3000x_1 + 2000x_2 \leq 16000$$
$$x_1 \leq 4$$
$$x_2 \leq 5$$
$$0.04x_1 + 0.06x_2 + d_1^- - d_1^+ = 0.32$$
$$0.072x_1 + 0.036x_2 + d_2^- - d_2^+ = 0.288$$
$$x_1 - 2x_2 + d_3^- - d_3^+ = 0$$
$$x_1, x_2 \quad \text{Integer} \quad \& \quad d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+ \geq 0$$

A detailed solution summary is given in Table 5.2., and a simplified version of the C programming code for this problem can be found in Appendix I.

Table 5.2 Solution summary for Example 2

Reported solution	**Goal satisfaction levels**	**Computation time**	**Explanations**
x_1=2, x_2=4	z_1=0, z_2=0, z_3=12	Not reported	Optimum
Tabu search solutions			
Parameter set: *Stepi*=3, *nneigh*=10 Solution: x_1=2, x_2=4	z_1=0, z_2=0, z_3=12	2.5 sec.	Optimum
Parameter set: *Stepc*=3, *nneigh*=5 Solution: x_1=2, x_2=4	z_1=0, z_2=0, z_3=12	2.5 sec.	Optimum
Parameter set: *Stepc*=1, *nneigh*=7 Solution: x_1=2, x_2=4	z_1=0, z_2=0, z_3=12	4 sec.	Optimum
Parameter set: *Stepc*=1, *nneigh*=4 Solution: x_1=2, x_2=4	z_1=0, z_2=0, z_3=12	4 sec.	Optimum

As can be seen from the results, in all parameter settings Tabu search found the optimal solution for this problem. It may also be possible to say that Tabu search is less sensitive to the setting of parameters for integer variables than continuous ones.

Example 3

The following model is taken from De *et. al.*(1982)'s paper for optimally solving a capital budgeting problem. They used an implicit enumeration methodology to obtain the solution for the zero-one linear pre-emptive goal programming problem.

$$\text{lexmin}\{z_1 = (d_2^+), z_2 = (d_1^-), z_3 = (d_7^+), z_4 = (d_3^- + d_4^- + d_5^- + d_6^- + d_8^+)\}$$

s.t.

$$45.48x_1 + 37.32x_2 + 47.47x_3 + 30.23x_4 + 31.37x_5 + d_1^- - d_1^+ = 110.55$$

$$150x_1 + 120x_2 + 90x_3 + 20x_4 + 80x_5 + d_2^- - d_2^+ = 250$$

$$66.32x_1 + 48.37x_2 - 41.17x_3 - 30x_4 - 40x_5 + d_3^- - d_3^+ = 4.95$$

$$58.13x_1 + 58.13x_2 + 48.13x_3 - 30x_4 + 38.61x_5 + d_4^- - d_4^+ = 6.05$$

$$58.13x_1 + 39.20x_2 + 87.66x_3 + 72.12x_4 + 29.20x_5 + d_5^- - d_5^+ = 7.56$$

$$58.70x_1 + 49.24x_2 + 96.72x_3 + 68.80x_4 + 29.24x_5 + d_6^- - d_6^+ = 7.92$$

$$105.5x_1 + 83.4x_2 + 92.4x_3 + 46.8x_4 + 54.1x_5 + d_7^- - d_7^+ = 300$$

$$1.086x_1 + 1.624x_2 + 0.946x_3 + 0.370x_4 + 0.438x_5 + d_8^- - d_8^+ = 3.8$$

$$x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1$$

$$d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+, d_5^-, d_5^+, d_6^-, d_6^+, d_7^-, d_7^+, d_8^-, d_8^+ \geq 0$$

A detailed solution summary for this problem is given in Table 5.3. The step size for zero-one variables is fixed, therefore in this application the number of neighbourhood solutions and the size of the Tabu list are varied in order to observe their effects on the solution quality.

Table 5.3 Solution summary for Example 3

Reported solution	**Goal satisfaction levels**	**Computation time**	**Explanations**
x_1=0, x_2=1, x_3=1, x_4=1, x_5=0	z_1=0, z_2=0, z_3=0, z_4=27.75	Not reported	Optimum
Tabu search solutions			
Parameter set: *Nneigh*=20, *m*=10, Solution: x_1=0, x_2=1, x_3=1, x_4=1, x_5=0	z_1=0, z_2=0, z_3=0, z_4=27.75	10.5 sec.	Optimum
Parameter set: *Nneigh*=10, *m*=10, Solution: x_1=0, x_2=1, x_3=1, x_4=1, x_5=0	z_1=0, z_2=0, z_3=0, z_4=27.75	6.2 sec.	Optimum
Parameter set: *Nneigh*=5, *m*=20, Solution: x_1=0, x_2=1, x_3=1, x_4=1, x_5=0	z_1=0, z_2=0, z_3=0, z_4=27.75	4 sec.	Optimum
Parameter set: *Nneigh*=3, *m*=15, Solution: x_1=0, x_2=1, x_3=1, x_4=1, x_5=0	z_1=0, z_2=0, z_3=0, z_4=27.75	3.8 sec.	Optimum

As can be seen from the results, in all parameter settings Tabu search found the optimal solution. It may also be possible to say that Tabu search is less sensitive to the setting of parameters for zero-one variables than continuous and integer variables.

Many other test problems have also been solved successfully. Some of these test problems are given in Appendix II. Comparisons were also made with LINDO™ (LINDO, 1996) software for solving linear and integer goal programming problems. In general, there is not a significant difference between computational times (computational time in Tabu search depends on the determination of tabu search parameters), but the quality of the solutions with the present Tabu search algorithm

were about 10%-35% better, especially with continuous variables. However a more intensive research may be useful in determining the effect of setting Tabu search parameters on the quality of solutions obtained and the computational time requirements.

5.2.4 Application of the proposed algorithm to simulation optimisation

As explained in the previous sections, a typical optimisation problem is to maximise (minimise) real valued function(s) where feasible points are restricted to some constraint set. If this problem is analytically intractable, as is often the case in manufacturing applications, then the decision-maker may choose to simulate the system. The optimisation of all or some objective functions which cannot be stated (or at least easily) analytically is the objective of simulation optimisation. In other words, the aim is to determine the inputs, or parameters, which optimise the outputs, or objective functions, of the simulation experiment. Note that the values of the simulation outputs are actually realisations of a random variable because of the random nature of the process described by the inputs. In addition, when an objective function is defined using the simulation outputs, it also becomes stochastic in nature. For the purposes of optimisation, the simulation is simply a black box that computes a realisation of the function value for a given combination of the parameter values.

Optimisation strategies in many of the classic optimisation techniques require mathematical operations (i.e. derivative, integration etc.) (Michalewicz, 1996). Thus, they require analytical forms of the objective functions and constraints. Therefore they cannot be applied directly to simulation optimisation. Additionally, they are

generally applicable to convex functions and cannot find global optimum solutions easily. As its counterparts simulated annealing and genetic algorithms, tabu search, and its extension multiple objective tabu search as presented here, is a heuristic technique developed to solve complex optimisation problems. It does not depend on the form of the objective functions and constraints and does not perform mathematical operations like taking derivatives or integration, etc. It works with solution strings and only requires the performance of each solution which can be determined from an analytical equation (if available) or from a simulation model. The optimisation strategy is the same and does not depend on the evaluation strategy of objective functions. So the proposed TS based MOO algorithms can readily be used for simulation optimisation or hybrid analytic/simulation optimisation. The flowchart shown in Figure 5.2 depicts the simulation optimisation strategy applied in this research work.

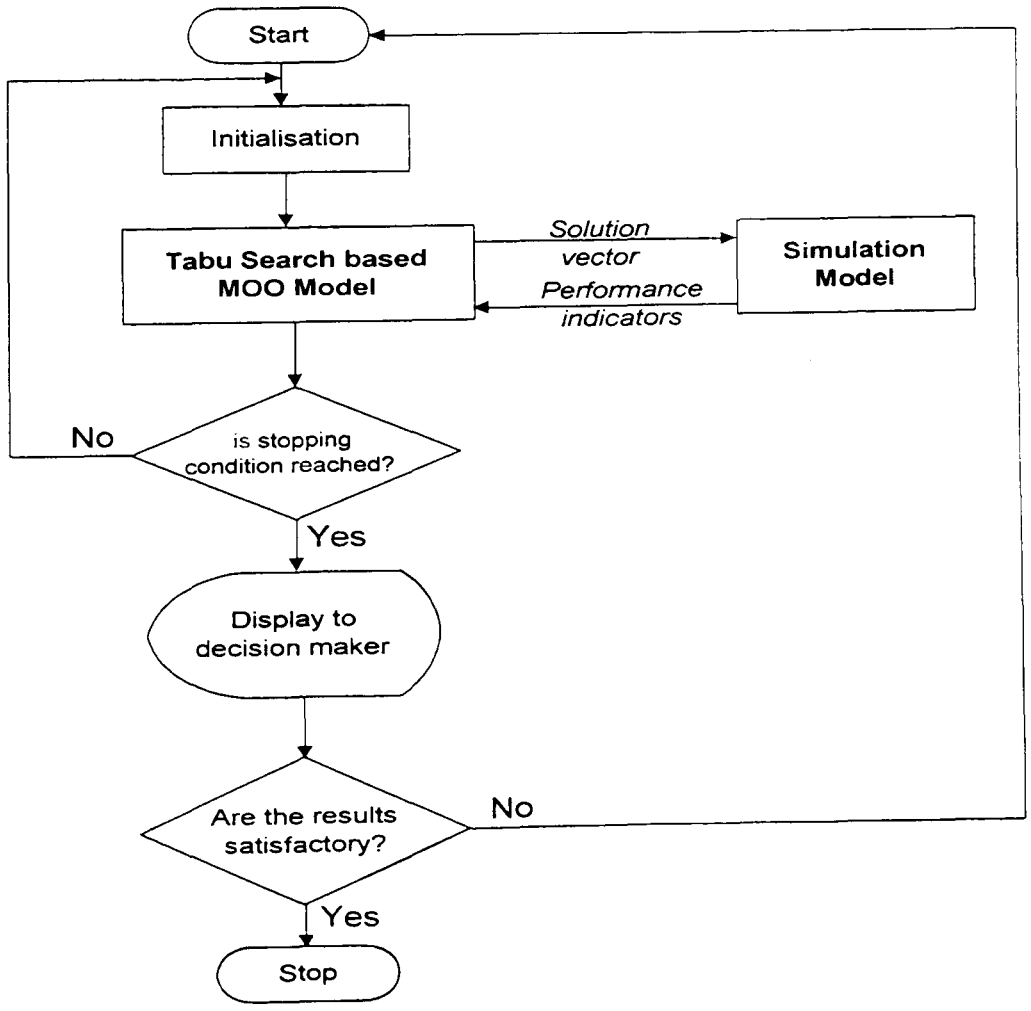


Figure 5.2 The flowchart of simulation optimisation strategy

5.3 CONCLUSIONS

The proposed TS algorithm to solve pre-emptive goal programming models has been explained in detail in this chapter. A detailed comparative study has also been carried out. The proposed algorithm is also the first application of TS to solve pre-emptive goal programming models. It has been observed that TS is a promising candidate to solve pre-emptive goal programming models. In every application the solutions found were at least as good, if not better than, the reported results. In some applications 10-35% improvement in solution quality has been obtained. The application of TS to multiple objective simulation optimisation problems is also explained in this chapter. TS is also a good candidate for multiple objective simulation optimisation problems due to its problem-independent nature.

CHAPTER SIX

6. DEVELOPMENT OF MULTIPLE OBJECTIVE MANUFACTURING CELL FORMATION MODULE

6.1 INTRODUCTION

In this chapter, the configuration module of the framework is explained. A new approach is proposed to simultaneously form independent part/machine cells. The cell formation strategy is based on *capability/requirement* analysis. Therefore, it can be named as a *capability based approach* for configuring cellular manufacturing systems (CMS). A part list that contains Resource Elements (RE) based generic part process plans constitutes the main input for the cell formation model. The capabilities of available machine tools are also defined in terms of REs. It is possible to consider overlapping capabilities of machines by using RE definition while forming manufacturing cells. Therefore, the opportunity of forming independent manufacturing cells increases. A processing requirement in the operation sequence of a part can be processed on any machine that can satisfy its RE requirements. Consequently, all process planning options of parts can be automatically considered for independent cell formation decisions.

Manufacturing cells are formed by matching capabilities of existing machines with the RE based processing requirements of parts. While forming the manufacturing cells the following issues are concerned:

- Formed cells sizes should be within the user-defined limits. Cell sizes are generally constrained in order to have proper control on cells and reduce the scheduling complexity. However there are no specific known limits on cell sizes, because cell size is generally problem dependent and set by the designer in relation to some other design requirements and constraints (i.e. material handling restrictions due to robots, scheduler's capability etc.) (Gallagher and Knight, 1986, Kusiak, 1990).
- Formed cells should be self-sufficient i.e. enough capacity in each cell and no (or minimum) inter-cell movement. Cell independence is the main objective of cellular manufacturing applications (Burbidge, 1987). An increase in cell interactions generally causes an increase in controlling complexity, increase in material handling cost, decrease in part quality etc. Therefore it should be avoided if possible.
- Formed cells should be able to offer flexibility to handle changing production requirements. Due to rapid changes in product requirements, especially in today's market, flexibility should be considered in the design (re-design) of manufacturing cells (Dahel and Smith 1993, Vakharia and Kaku, 1994).
- Parts in the same cell should be similar in terms of processing requirements and operation sequences. If parts in the formed cells follow similar processing routes, the following benefits are likely to occur: reduction in intra-cell material handling cost, easiness in automation, easiness in process control, increased part quality, etc. (Tam, 1990).
- Load should be balanced between formed cells in order to prevent heavy utilisation of some cells and lower utilisation of others. This objective is

important for enhancing operating efficiency of formed cells (Han and Ham, 1989).

All the above issues are given major importance by Wemmerlov *et. al.*,(1987, 1989, 1997) and Vakharia and Wemmerlov (1990) for a successful cell formation practice. The cell formation problem is formally stated as a pre-emptive goal programming model and solved by the multiple objective tabu search algorithm which has been introduced in Chapter 5. A comparative study is also done to present the efficiency of the proposed approach.

6.2 MATHEMATICAL FORMULATION OF THE MULTIPLE OBJECTIVE CELL FORMATION PROBLEM

Formation of the part/machine cells is formally stated as a pre-emptive goal programming model. The main characteristics of the model are as follows:

- It simultaneously forms part/machine cells for cellular manufacturing applications. As discussed in Chapter 2-Section 2.5 simultaneous cell formation approach is advantageous.
- It minimises dissimilarity between parts in each formed cell.
- It minimises total load unbalance between formed cells.
- It minimises extra capacity requirements while converting an existing job shop to a cellular shop. To achieve the main goal of cellular manufacturing (i.e. forming independent manufacturing cells) extra copies of some machines might be required while forming the manufacturing cells in order to prevent inter-cell movements resulting from capability or capacity inadequacies (Adil *et. al.*, 1996).

However, extra capacity requirements must be minimised when possible to minimise the costs of implementing CMSs.

- It maximises flexibility of formed cells via maximising the number of different REs in each formed cell.
- Maximum and minimum number of parts and machines in each cell and cell independence are considered as the constraint functions.
- Capabilities of machines in the facility and processing requirements of parts (as generic process plans) are both defined in terms of REs in the model.
- The total number of cells is defined by the decision-maker.

The following notation is used in the mathematical model:

Indexes

i,l :Part indexes.

j : Machine index.

k : Cell index.

r : RE index.

Parameters

m : Number of parts.

n : Number of machines.

g : Number of cells.

nRe : Number of REs.

DS_{il} : Dissimilarity between parts i and l.

CL_k : Load on cell k.

ACL : Average cell load.

H_j : Capacity of machine j (minutes/year).

Q_i : Demand for part i.

t_{ir} : Processing time for RE r in part i (minutes).

M_{min} : Minimum number of machines in a cell.

M_{max} : Maximum number of machines in a cell.

P_{min} : Minimum number of parts in a cell.

P_{max} : Maximum number of parts in a cell.

$FEMIX_{kr}$: 1 if RE r is available in cell k, 0 otherwise.

p_{jr} : 1 if RE r is available on machine j, 0 otherwise

q_{ir} : 1 if RE r is required by part i, 0 otherwise

w_1, w_2 : Weights of part similarity equation.

Variables

Y_{jk} : 1, if machine j is in cell k, 0 otherwise.

X_{ik} : 1, if part i is in cell k, 0 otherwise.

d_{ϕ}^{-}, d_{ϕ}^{+} : Under and over achievement of *'dissimilarity goal'*.

d_{θ}^{-}, d_{θ}^{+} : Under and over achievement of *'cell load unbalance goal'*.

d_{ε}^{-}, d_{ε}^{+} : Under and over achievement of *'flexibility goal'*.

d_{k}^{-}, d_{k}^{+} : Shows the amount of deviation from the total available capacity in cell k.

d_{rk}^{-}, d_{rk}^{+} : Shows the amount of deviation from the total capacity available for RE r in cell k.

Mathematical formulation

(1) *Goal structure*

In the mathematical modelling of the cell formation problem, objective functions are described as goal constraints that are explained in the following paragraphs. The purpose of the optimisation is to minimise the deviations from these goals by considered highly prioritised goals first. Goals could be ordered according to their relative importance. The result of this ordering process is a goal structure. The goal structure will differ depending on the situation and the preferences of the designer. Equation 6.1 represents the lexicographical order of deviational variables to be minimised.

$$\text{Lexmin}\{(d_{\phi}^{+}),(\sum_{k=1}^{g}\sum_{r=1}^{nRE}(d_{k}^{-}+d_{rk}^{-}),(d_{\theta}^{-}+d_{\theta}^{+}),(d_{\varepsilon}^{-}))\} \qquad 6.1$$

In equation 6.1, dissimilarity goal is selected as the most important goal to be achieved. Capacity goal, cell load unbalance goal and flexibility goal are the 2nd 3rd and 4th important goals to be achieved. However, the developed cell formation system lets the designer to change the order of goals if required.

(2) *Dissimilarity goal constraint*

Part similarity (or dissimilarity) can be characterised by two dimensions: a) commonality in machine requirements, and b) similarity patterns of production sequences. Although both of them are important, GT research has tented to focus its attention primarily on the former. In this study both of them are considered and given equal importance. Part dissimilarity based on commonality of machine requirements is determined by using the following equation: $PDS_{il} = 1 - (\mathbf{P}_i \cap \mathbf{P}_l \,/\, \mathbf{P}_i \cup \mathbf{P}_l)$. Where, PDS_{il} is dissimilarity level between parts i and l based on RE requirements, numerator shows number of common RE between parts i and l, and denominator shows total number of RE requirements for parts i and l. Part dissimilarity (SDS_{il}) level between parts i and l which is based on RE based generic processing sequences is determined by using a dynamic programming procedure proposed by Tam (1990). The procedure is given as follows:

```
Input:
        P_i = {RE1,RE2.....REn}  /* operation sequence for part i  */
        P_l = {RE1,RE2.....REm} /* operation sequence for part l  */

Output:
        SDS_{i,l}

Algorithm:
step_1→ Create an n*m matrix D;
step_2→ Initialise D[0][0]=0;
step_3→ Initialise the first row (D[0][k],  0<=k<=n) to be the sequence 0,1,2,.....n;
step_4→ Initialise the first column (D[j][0], 0<=j<=m) to be the sequence 0,1,2,....m;
step_5→ for( k=1; k=n; k++)
        {
            for( j=1; j=m; j++)
            {
                if (P1[k] = = P2[j])
                        substitute = D[k-1][j-1];
                else
                    substitute = D[k-1][j-1]+1;
                    delete = D[k-1][j]+1;
                    addition = D[k][j-1]+1;
                D[k][j] = min(substitute, delete, addition);
            }
        }
step_6→ SDS_{i,l} = D[n,m];
```

The overall dissimilarity level (DS_{il}) between two parts i and l is defined as the weighted sum of the above dissimilarity indices that is calculated by the following equation;

$$DS_{il} = w_1 * PDS_{il} + w_2 * SDS_{il} \qquad 6.2$$

Where, w_1 and w_2 are weights associated with each dissimilarity index. In equation 6.2, $w_1+w_2=1$ and $w_1, w_2>=0$.

Dissimilarity goal constraint that is given by equation 6.3 determines deviations from the dissimilarity goal in formed cells. Dissimilarity goal is set to zero in equation 3 (i.e. the target is to obtain %0 part dissimilarity in each formed cell).

$$\sum_{k=1}^{g} \sum_{i=1}^{m} \sum_{l=1}^{m} DS_{il} X_{ik} X_{lk} + d_{\phi}^{-} - d_{\phi}^{+} = 0 \qquad 6.3$$

(3) *Capacity goal constraints*

In this work, operation sequences of parts are not defined based on machine requirements (i.e. machine routes are not known initially), an operation of a part can be executed by any machine which can satisfy the corresponding RE requirement in the processing sequence. Therefore standard machine capacity-requirement equations proposed in the literature cannot be applied (Co and Araar 1988, Moon *et. al.* 1997, Cheng *et. al.* 1996, Sankaran 1990, Gunasingh and Lashkari 1989, Dahel and Smith 1993, Wei and Gaither 1990, Rajamani *et. al.* 1992) because without knowing the specific machine routes, total load on the corresponding machines cannot be determined. In fact, parts may have many alternative machine routes in a cell and they can be re-routed for some reasons (e.g. machine breakdown etc.). Additionally, classical machine capacity equations are not sensitive to overlapping capabilities between machines and their usage may results excess capacity in cells by calling for the addition of extra machines. In this work, instead of focusing on individual machine capacities, the overall capacities of cells are considered and modelled with transportation-like equations below.

$$\sum_{j=1}^{n} Y_{jk} H_j - \sum_{i=1}^{m} \sum_{r=1}^{nRE} X_{ik} q_{ir} t_{ir} Q_i + d_k^- - d_k^+ = 0 \quad \forall k \qquad 6.4$$

$$\sum_{j=1}^{n} Y_{jk} p_{jr} H_j - \sum_{i=1}^{m} X_{ik} q_{ir} t_{ir} Q_i + d_{rk}^- - d_{rk}^+ = 0 \quad \forall r, k \qquad 6.5$$

Equation 6.4 calculates the deviation from the total available capacity in each cell[1]. In ideal conditions deviations are equal to zero (i.e. satisfaction of the *capacity goal*).

[1] Total capacity for a machine is defined in terms of total number of minutes a machine is available for part processing in the entire planning horizon. However other time units can also be used. But the same unit should be used for all time-related data (i.e. part processing time etc.). Capacity of a cell is equal to summation of capacities of its machines.

This means that there is potentially no excess capacity in the system (i.e. total capacity in each cell is equal to total processing time requirement from each cell). Negative deviation means that potentially available capacity in the cell is not sufficient (i.e. total capacity in some cells is less than total processing time requirements). However, in some situations, although the total potential capacity seems enough (i.e. total capacity in some cells is higher than total processing time requirements, but total capacity in any cell is not less than total processing time requirement), there may still be insufficient capacity for a number of processing capabilities (REs) in a cell (i.e. all the parts assigned to a cell demands too much processing for an individual operation but only a few machines have the required capability, in such cases although total capacity of the cell seems enough there will be capacity problems).

Deviations from total available capacity for individual processing capability units (RE) in each cell are calculated by using equation 6.5. A positive deviation means that there is enough capacity for the corresponding capability unit in the cell for the corresponding processing requirement of the parts assigned to the cell. If the summation of the negative deviations in equations 6.4 and 6.5 are equal to zero, then the capacity in the corresponding cells is certainly enough to satisfy the total processing requirements of the corresponding parts assigned to them. An explanatory table is given in figure 6.1 that shows working of equations 6.4 and 6.5.

RE-1 is not available on machine 3

Total processing time requirement for RE-4 from the parts that assigned to Cell 1

	$\sum t_{i1}$=3000	$\sum t_{i2}$=2000	$\sum t_{i4}$=5000	$\sum t_{i1}+\sum t_{i2}+\sum t_{i4}$=10000
Y_{11}=1	4000	*	4000	H_1=4000
Y_{31}=1	*	3500	*	H_3=3500
Y_{41}=1	3000	3000	*	H_4=3000
	d_{11}^+=4000	d_{21}^+=4500	d_{41}^-=1000	$H_1+H_3+H_4$=10500 d_1^-=500

Capacity of Machine 4

Machine 4 is in Cell 1

Negative deviation
1000 units extra capacity is required for capability unit 4 (i.e. RE-4) in Cell 1

Positive deviation
There is no capacity shortage for capability unit 1 (i.e. RE-1) in cell 1

Total potentially available capacity in Cell 1seems enough (i.e. positive deviation), however there is a capacity shortage in Cell 1 because not all negative deviations are equal to zero i.e. there is an extra 1000 units capacity requirement for RE-4.

Figure 6.1 Analysis of capacity deviations in a cell

The following useful data can be obtained from the deviational variables; total extra capacity requirements in a cell that should be satisfied (d_k^-), total excess capacity in a cell (d_k^+), total potentially excess capacity for a particular capability unit in a cell (d_{rk}^+), (it is not possible to make use of all potentially excess capacity of a particular capability unit, if it is greater than d_k^+), total extra capacity requirement for a particular capability unit in a cell that should be satisfied (d_{rk}^-). In the case of extra capacity requirements while reconfiguring from a job shop to a cell shop, the proposed model determines which specific capability units are required after considering all overlapping capabilities between machines. Therefore, it is possible to invest in the correct machines that have just the required capabilities.

Consequently, money can be saved by reducing the effects of the non-integer machine requirements problem (Sule, 1991). Additionally, capacity utilisation levels can be increased by this approach which is known as a problem in cellular manufacturing applications.

(4) *Cell load unbalance goal constraint*

Load should be balanced between formed cells (i.e. unbalance should be minimised) in order to prevent heavy utilisation of some cells and lower utilisation of others. This objective is important for enhancing operating efficiency of formed cells (Han and Ham, 1989). Equation 6.6 represents the cell load unbalance goal constraint in the present model.

$$\sum_{k=1}^{g} (CL_k - ACL)^2 / g + d_{\theta}^{-} - d_{\theta}^{+} = 0 \qquad 6.6$$

In equation 6.6, ACL is average utilisation of the cells that is calculated by using equation 6.7:

$$ACL - (\sum_{k=1}^{g} CL_k) / g = 0 \qquad 6.7$$

Utilisation of each cell CL_k, is calculated by using equation 6.8;

$$CL_k - (\sum_{i=1}^{m} \sum_{r=1}^{nRE} X_{ik} q_{ir} t_{ir} Q_i / \sum_{j=1}^{n} Y_{jk} H_j) = 0 \quad \forall k \qquad 6.8$$

(5) *Flexibility goal constraint*

Formed cells should be able to offer flexibility to handle changing production requirements. Due to rapid changes in product requirements, especially in today's market, flexibility should be considered in the design of manufacturing cells (Dahel and Smith 1993, Vakharia and Kaku 1994). In this work, cell flexibility is measured by counting the number of different REs available in a cell. This definition is similar

to the definitions given by Dahel and Smith (1993), Vakharia and Kaku (1994). But they considered the total number of different machine types available in cells. In fact, it may not be possible to assign all machine types to each cell because of economic and control reasons (i.e. cell size). But, it may be possible to design cells in such a way that availability of different capability units in each cell is maximised. By employing this approach, flexibility of cells can be increased without increasing their size beyond the specified limits. Equation 6.9 represents the cell flexibility goal constraint.

$$\sum_{k=1}^{g}\sum_{r=1}^{nRE} FEMIX_{kr} + d_{\varepsilon}^{-} - d_{\varepsilon}^{+} = nRE * g \qquad 6.9$$

In equation 6.9, cell flexibility goal is equated to *nRE*g* which means we want to make all RE available in each cell (i.e. maximum flexibility). The total number of REs in each cell is determined by using equation 6.10:

$$FEMIX_{kr} - \sum_{j=1}^{n} (Y_{jk}\, p_{jr}) / ((Y_{jk}\, p_{jr}) + \kappa) = 0 \quad \forall k, r \qquad 6.10$$

where; $\kappa = \begin{cases} 1 & if \quad \sum_{j} Y_{jk}\, p_{jr} = 0 \\ & \text{else} \\ 0 & \end{cases}$

(6) *Hard constraints*

In a goal programming model hard constraints shape the feasible search space. Satisfaction of hard constraints is compulsory.

The first set of hard constraints (equations 6.11 and 6.12) ensure that each part and machine are assigned to only one cell.

$$\sum_{k=1}^{g} X_{ik} - 1 = 0 \quad \forall i \tag{6.11}$$

$$\sum_{k=1}^{g} Y_{jk} - 1 = 0 \quad \forall j \tag{6.12}$$

Size of the manufacturing cells is also hard constrained in the proposed cell formation model. In order to have proper control on cells and reduce the scheduling complexity cell sizes are generally constrained in cell formation applications. However there is not known limits and a consensus on cell sizes. Because cell size is generally problem dependent and set by the designer in relation to some other design requirements and constraints (i.e. material handling restrictions due to robots, scheduler's capability etc.) (Gallagher and Knight 1986). In the proposed cell formation model cell size constraints are given by equations 6.13-6.16.

$$\sum_{i=1}^{m} X_{ik} - P_{\min} \geq 0 \quad \forall k \tag{6.13}$$

$$\sum_{j=1}^{n} Y_{jk} - M_{\min} \geq 0 \quad \forall k \tag{6.14}$$

$$P_{\max} - \sum_{i=1}^{m} X_{ik} \geq 0 \quad \forall k \tag{6.15}$$

$$M_{\max} - \sum_{j=1}^{n} Y_{jk} \geq 0 \quad \forall k \tag{6.16}$$

Constraints 6.13 and 6.14 ensure that, number of parts and machines in each cell should not be lower than the specified limits. Constraints 6.15 and 6.16 ensure that, the number of parts and machines in each cell should not be higher than the specified limits.

The last hard constraint is related to the capability of the formed cells. Constraint 6.17 ensures that processing requirements of part families can be totally satisfied by the machines in their cells.

$$\sum_{j=1}^{n} Y_{jk} p_{jr} + \gamma - \sum_{i=1}^{m} X_{ik} q_{ir} > 0 \quad \forall r, k \qquad 6.17$$

In equation 6.17, $\gamma = \begin{cases} 0 & if \quad \sum_{j} Y_{jk} p_{jr} = 0 \\ \text{else} \\ \text{A big number} \end{cases}$

$$X_{ik}, Y_{jk} \in \{0,1\}$$

$$d_{\phi}^{-}, d_{\phi}^{+}, d_{\theta}^{-}, d_{\theta}^{+}, d_{\varepsilon}^{-}, d_{\varepsilon}^{+}, d_{k}^{-}, d_{k}^{+}, d_{rk}^{-}, d_{rk}^{+} \geq 0 \ \forall i, j, k, r \qquad 6.18$$

Constraint 6.18 ensures integrality.

(7) *Model size*

The final explanation for the proposed model is about its size, $(m+n)*g$ zero-one variables are needed for a problem of n machines, m parts, and g cells. The number of zero-one variables of this model is far fewer than that of the famous *p-median* formulation (Kusiak, 1987), which is $m*m$. It should also be mentioned that, *p-median* model is developed to find the part families or the machine cells only. For example, a problem with 50 parts, 20 machines and 4 cells requires 280 zero-one variables in the present model, whereas the *p-median* model needs 2500 zero-one variables which is almost eight times more. This feature is especially important for solving big size cell formation problems within a reasonable computation time.

6.3 APPLICATION OF MULTIPLE OBJECTIVE TABU SEARCH ALGORITHM TO SOLVE THE MATHEMATICAL MODEL

The multiple-objective Tabu Search (TS) algorithm that has been developed in Chapter 5 is employed to solve the above mathematical model. However, a number of problem-specific features are incorporated into the original TS algorithm. The original TS algorithm can also be used without any change. TS is a flexible technique and incorporation of some problem specific features (if possible) may improve its speed while converging to a solution (Glover, 1990,1993).

The following problem related enhancements have been made to speed up the original multiple objective TS algorithm:

Generation of the neighbourhood solutions:

Instead of using the neighbourhood generation functions that were presented in Chapter 4, Section 4.2.1., the following strategy is employed for the generation of neighbourhood solutions:

- Randomly select a machine from a randomly determined cell whose total number of machines is not less than or equal to M_{min}. Assign this machine to a randomly selected cell whose total number of machines is not more than or equal to M_{max}.
- Randomly select a part from a randomly determined cell whose total number of parts are not less than or equal to P_{min}. Assign this part to a randomly selected cell whose total number of parts are not more than or equal to P_{max} and can satisfy all processing requirements of the part.
- Apply the above steps to generate a previously determined number of neighbourhood solutions.

By employing this strategy one can easily generate feasible neighbourhood solutions that satisfy all hard constraints (Equations 6.12-6.18). Figure 6.2 depicts a solution vector and generation of 3 neighbourhood solutions from it for a (4 parts, 4 machines, 2 cells) hypothetical cell formation example.

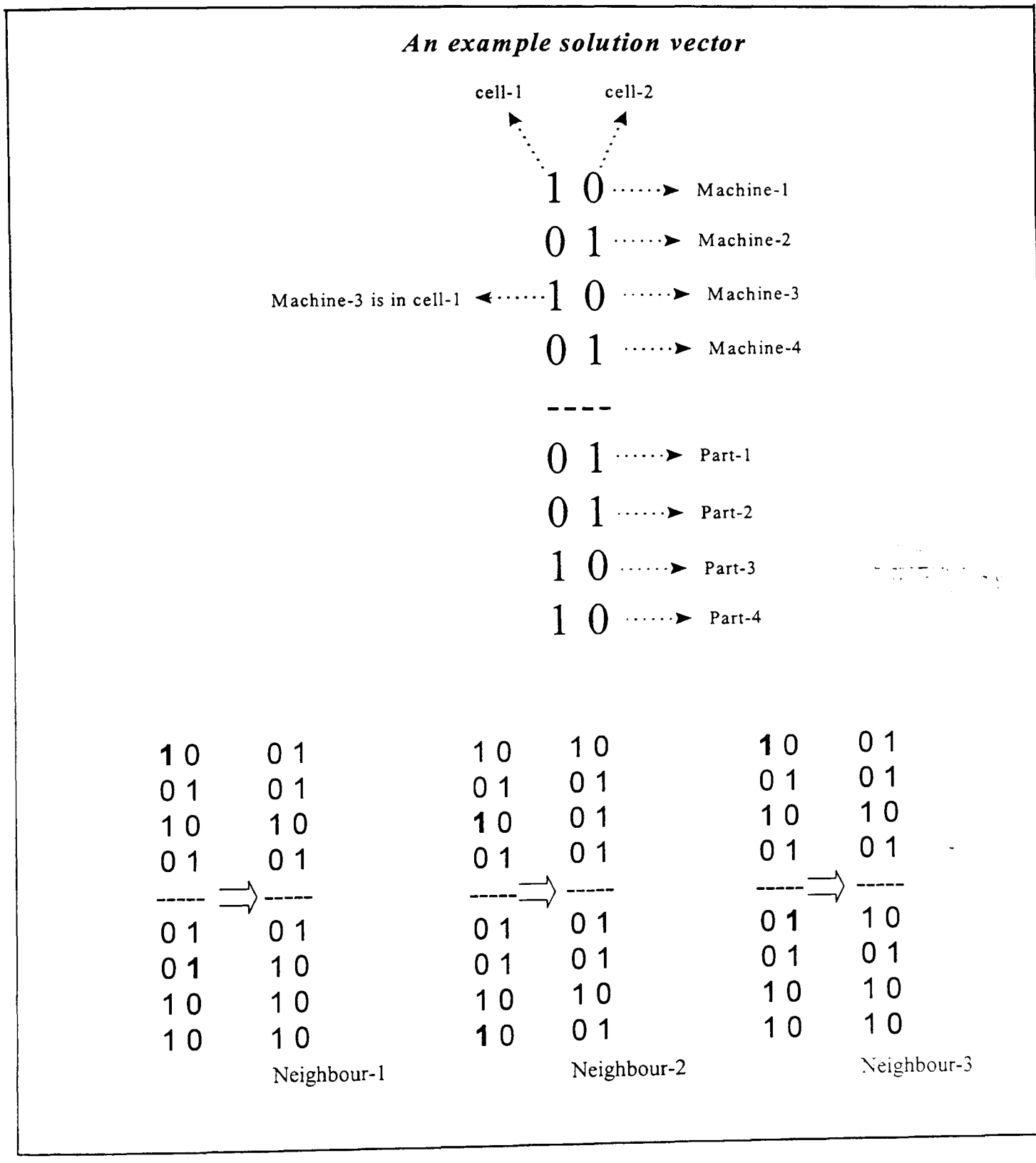


Figure 6.2 Neighbour solutions generation (randomly selected part and machine are bolded)

Tabu list:

Instead of putting the whole solution vectors in the *tabu list* as tabu solutions (see Chapter 5-Section 5.2.2) only indices of randomly selected and reassigned part-machine pairs are put into the *tabu list*.

Aspiration criteria:

Only the indices of part-machine pairs as the *solution features* are put into the *tabu list*. However, reselection of these features in later iterations might generate different solution vectors, because these features themselves are not the solution vectors. Therefore global optimum solution can be missed if these solution features are directly considered as tabu. In order to prevent this situation in TS applications an *aspiration criterion* needs to be defined to override the tabu status of solution features when necessary (Glover, 1990).

The aspiration criterion is defined as follows: Any move that improves the *best known solution* (see Chapter 5-Section 5.2.2) is accepted, even if the move is tabu. If the whole solution vectors are put into the *tabu list,* then there is no need to define an *aspiration criterion*. However, in large problems this might not be favourable.

6.4 EXAMPLE APPLICATIONS AND THE COMPARATIVE WORK

The multiple objective TS algorithm to solve the above model is programmed in FORTRAN. Several test problems with various sizes have been solved with the developed computer program for testing the model and the efficiency of the developed algorithm. It takes approximately two minutes to obtain a good quality

solution for a medium-sized problem (i.e. 20 machines and 30 part types with 10 different REs).

The example application is performed on the test model that is used through this study and explained in the following chapter. It contains twelve machines and seven machine types. Machine tools available in the test job shop and part data for the example application are given in Tables 6.1 and 6.2 below:

Table 6.1 Machine tools and their capabilities based on REs in the test model

Machine Tools	# Copies	Resource Elements											Capacity *1000(min/ year)
		1	2	3	4	5	6	7	8	9	10	11	
1-Drill Press-1	1	*											68
2-MHP Machining Centre-1	3(a,b,c)	*	*	*				*	*	*	*	*	66
3-Colchester Lathe-1	2(a,b)	*	*		*			*					64
4-MHP MT50 NC Lathe-2	2(a,b)	*	*		*			*			*		64
5-CNC Grinding Machine-1	2(a,b)					*	*						64
6-Jones & Shipman Cyc. Grinder	1						*						65
7- Jones & Ship. Surf. Grinder	1					*							64

Table 6.2 Part data: processing time (min), total demand, processing sequences

Parts	RE1	RE2	RE3	RE4	RE5	RE6	RE7	RE8	RE9	RE10	RE11	Demand *1000	RE Based Operation Sequences[2]
1	6	8		9								3	RE1-2-4
2	5	6	4									1	1-2-3
3					2	6	8					2.5	5-6-7
4					4			5				1.52	8-5
5				5	6		8					1.48	7-4-5
6						5	5	6				3.5	8-6-7
7								7	7	8		1	8-9-10
8									6	5	9	2	9-10-11
9	7	5			3							3	5-1-2
10			4	5								2	3-4
11					5		5		4			4.5	5-6-9
12								7	8	8		1	10-8-9
13					1			3		5		3	5-8-10
14					5		4	5				2.5	8-7-5
15	6	2										2.5	1-2
16			6	3								1.9	3-4
17						4	5	7				2.4	6-7-8
18								5	5	5	2	1.2	8-9-10-11
19		5			5							1.3	5-2
20							1	5	3			3	7-8-9

[2] RE based operation sequence of parts represents the order in which REs need to be used.

The TS algorithm was iterated 1500 times with a neighbourhood size of 5 and tabu list size of 15, number of cells is 3, maximum and minimum number of parts and machines in each cell were defined as 9,4 and 5,3 respectively. In around 3 minutes of computational time the algorithm converged to a solution (on a Pentium-200 PC with 32MB RAM). All necessary data i.e. similarity matrices etc. are calculated by the computer program. The output summary of the developed computer program which was named as MOCACEF 1.0 (**M**ultiple **O**bjective **CA**pability based **CE**ll **F**ormation) is shown in Table 6.3 below;

Table 6.3 The output summary of MOCACEF 1.0

Cell Number	Machines	REs Availability	Parts	Operation Sequences	Cell Capacity Utilisation	Extra Capability Requirement	Comments
1	1 2a 7	4 and 6 are not available	4 7 8 12 13 18	8-5 8-9-10 9-10-11 10-8-9 5-8-10 8-9-10-11	0.74	NIL	
2	3a 5a 4a 2b	All REs are available	1 2 9 10 15 16 19	1-2-4 1-2-3 5-1-2 3-4 1-2 3-4 5-2	0.76	NIL	
3	2c 4b 5b 6 3b	All REs are Available	3 5 6 11 14 17 20	5-6-7 7-4-5 8-6-7 5-6-9 8-7-5 6-7-8 7-8-9	0.89	NIL	

The example comparisons are made with two very well known and widely accepted production flow analysis based techniques from the literature (Kusiak, 1987, Seifoddini and Wolfe, 1986). To be able to solve the example problem with these techniques, a *part-machine matrix* is required, as this is the main input for them. One such matrix is generated from the Tables 6.1 and 6.2. Obviously, there are many possible alternative part-machine matrices that can be derived. Only one such matrix is reported here for comparative purposes. The final part-machine matrix is shown in Table 6.4. In Table 6.4 (RE-1) of Part-1 is assigned to Machine-1 that can supply (RE-1), (RE-2 and RE-4) of Part-1 are assigned to Machine-3 that can supply (RE-2 and RE-4). Different assignments can also be done for Part-1 (which results different part-machine matrices) all other parts are matched with machines similarly. If more than one RE of a part is assigned to the same machine then processing times for these REs are summed in the part-machine matrix (for Part-1, processing times for RE-2 and RE-4 are summed in the part-machine matrix, i.e. the new processing time is 8+9=17, see Table 6.4). It is obvious that some valuable information (processing sequences, alternative machines etc.) is not visible in part-machine matrices.

Table 6.4 Part-machine matrix

M/P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6								7						6					
2		15		5		11	22	20		4	9	23	3	9	2	3	12	17		9
3	17		2		5					5						4			5	
4					8				5				5							
5			14						3		5									
6						6											4			
7				4	6								1	5					5	

The solution obtained from the *p-median* model of Kusiak (1987) is summarised in Tables 6.5, 6.6, and 6.7

Table 6.5 The initial solution obtained from the *p-median* model

	1	3	9	11	15	6	17	2	4	5	7	8	10	12	13	14	16	18	19	20
1	*		*		*															
5		*	*	*																
6						*	*													
2				*	*	*	*	*	*		*	*	*	*	*	*	*	*		*
3	*	*								*			*				*		*	
4			*							*					*					
7									*	*					*	*			*	

Table 6.6 The final solution for the *p-median* model after eliminating inter-cell movements

	1	3	9	11	15	6	17	2	4	5	7	8	10	12	13	14	16	18	19	20	Load *1000	Machine Req.	# Mach.
1	18		21		15																54	0.79	1
5		35	9	22.5																	66.5	1.04	2
2				40.5	5																45.5	0.69	1
3	51	5																			56	0.88	1
4			15																		15	0.23	1
6						21	9.6														30.6	0.47	1
2						38.5	28.8														67.3	1.02	2
2								15	7.6		22	40	8	23	9	22.5	5.7	20.4		27	200.2	3.03	4
3										7.4			10				7.6		6		31	0.48	1
4										11.84					15						26.84	0.41	1
7									6.08	8.88					3	12.5			6		36.46	0.57	1

In Table 6.6, the load on each machine is calculated by summing total processing time requirements from each machine i.e. Load on Machine-1 is 18+21+15=54 (as demand for each part is divided by 1000, load should be multiplied with 1000). The machine requirement in each cell for each machine type is found by dividing the total load on each machine type by its capacity i.e. number of type 1 machine required in Cell-1 is 54/68=0.79 (1 machine of type 1 is required).

Table 6.7 Solution summary for *p-median* model

Cell Number	Machines	REs Availability	Parts	Operation Sequences	Cell Capacity Utilisation	Extra Machine Requirement	Comments
1	1 2a 3a 4a 5a 5b	All REs are Available	1 3 9 11 15	1-2-4 5-6-7 5-1-2 5-6-9 1-2	0.61	NIL	
2	2b 6	4 and 5 are not available	6 17	8-6-7 6-7-8	0.5	1 more copy of machine 2 is required	Cell utilisation is calculated including extra copies of machine 2
3	2c 3b 4b 7	6 is not available	2 4 5 7 8 10 12 13 14 16 18 19 20	1-2-3 8-5 7-4-5 8-9-10 9-10-11 3-4 10-8-9 5-8-10 8-7-5 3-4 8-9-10-11 5-2 7-8-9	0.65	3 more copies of machine 2 are required	Cell utilisation is calculated including extra copies of machine 2

In Table 6.7 a, b, c represent copies of the same machine type. Cell capacity utilisation is calculated by dividing total load in a cell by its capacity. The capacity of a cell is equal to summation of capacities of its machines. As an example from Table 6.7 capacity utilisation in Cell-1 is (54+66.5+45.5+56+15)/(68+66+64+64+2*64)=0.61.

The solution obtained from Seifoddini and Wolfe (1986)'s method is summarised in Tables 6.8, 6.9 and 6.10.

Table 6.8 The initial solution obtained from Seifoddini and Wolfe (1986)'s method

	1	3	9	11	15	6	17	2	4	5	7	8	10	12	13	14	16	18	19	20
1	*		*		*															
5		*	*	*																
6						*	*													
2				*	*	*	*	*	*		*	*	*	*	*	*	*	*		*
3	*	*								*			*				*		*	
4			*							*					*					
7									*	*					*	*			*	

Table 6.9 Final solution for Seifoddini and Wolfe (1986)'s method

	1	3	9	11	15	6	17	2	4	5	7	8	10	12	13	14	16	18	19	20	Load *1000	Machine Req.	# Mach.
1	18		21		15																54	0.79	1
5		35	9	22.5																	66.5	1.04	2
2				40.5	5																45.5	0.69	1
3	51	5																			56	0.88	1
4			15																		15	0.23	1
6						21	9.6														30.6	0.47	1
2						38.5	28.8	15													82.3	1.25	2
2									7.6		22	40	8	23	9	22.5	5.7	20.4		27	185.2	2.81	3
3										7.4			10				7.6		6		31	0.48	1
4										11.84					15						26.84	0.41	1
7									6.08	8.88					3	12.5			6		36.46	0.57	1

Table 6.10 Solution summary for Seifoddini and Wolfe (1986)'s method

Cell Number	Machines	REs Availability	Parts	Operation Sequences	Cell Capacity Utilisation	Extra Machine Requirement	Comments
1	1 2a 3a 4a 5a 5b	All REs are Available	1 3 9 11 15	1-2-4 5-6-7 5-1-2 5-6-9 1-2	0.61	NIL	
2	2b 6	4 and 5 are not available	6 17 2	8-6-7 6-7-8 1-2-3	0.57	1 more copy of machine 2 is required	Cell utilisation is calculated including extra copies of machine 2
3	2c 3b 4b 7	6 is not available	4 5 7 8 10 12 13 14 16 18 19 20	8-5 7-4-5 8-9-10 9-10-11 3-4 10-8-9 5-8-10 8-7-5 3-4 8-9-10-11 5-2 7-8-9	0.72	2 more copies of machine 2 are required	Cell utilisation is calculated including extra copies of machine 2

Results of the comparative study are depicted in Figures 6.3 and 6.4.

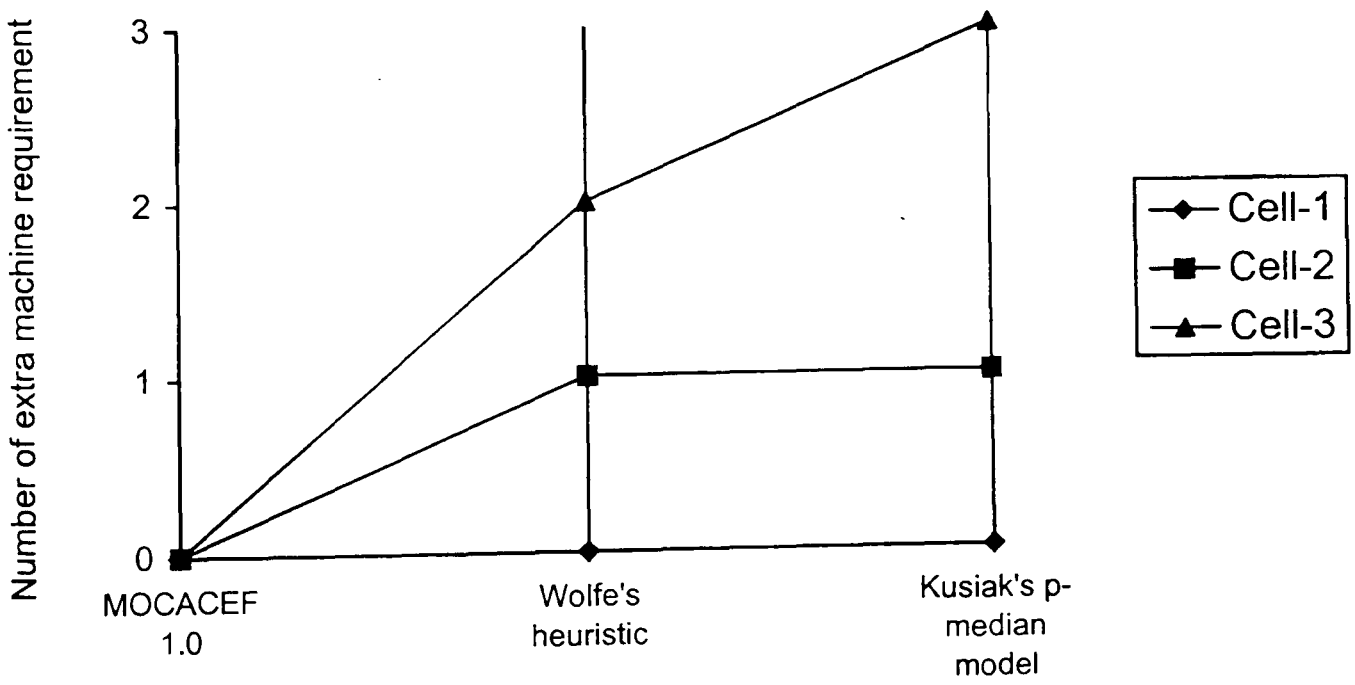


Figure 6.3 Comparison of extra resource requirement to configure cells

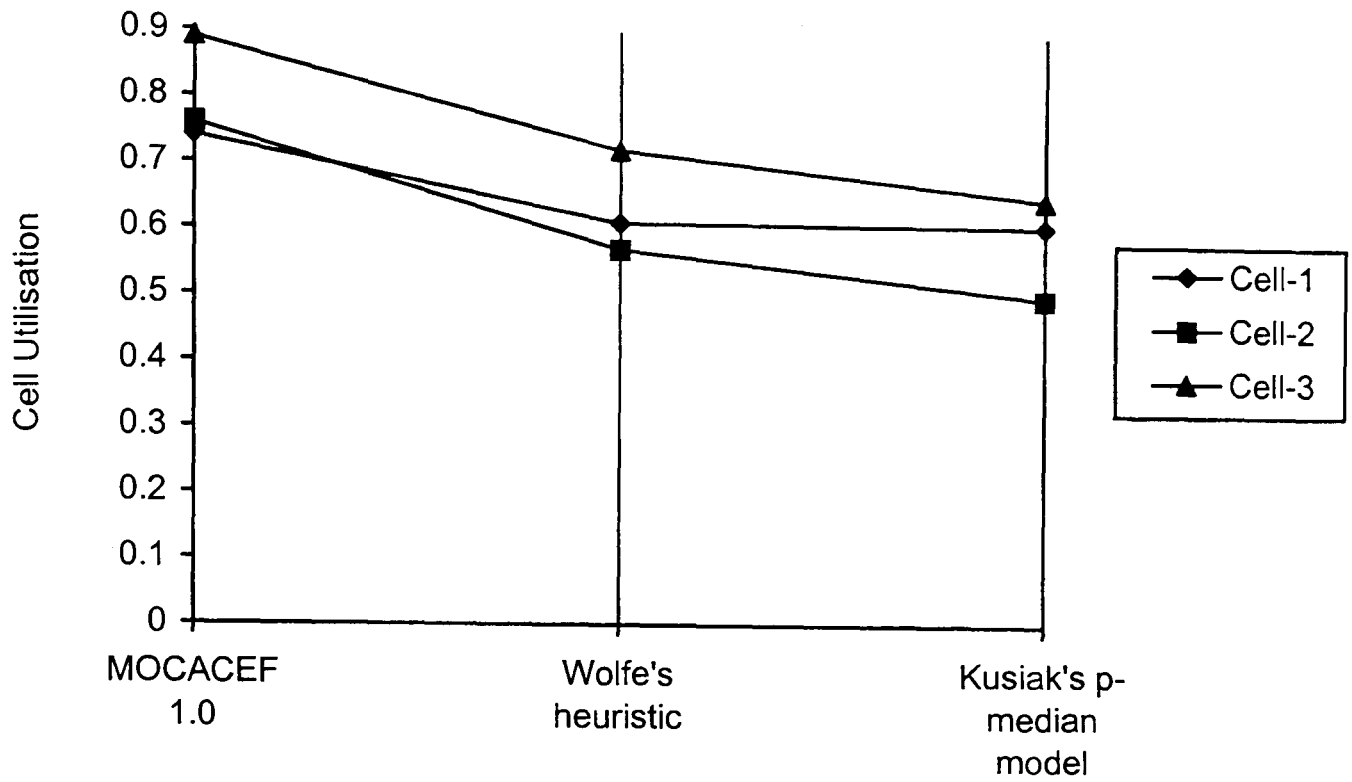


Figure 6.4 Comparison of cell utilisation levels

The proposed approach is also applied to other example problems. A case study with real data is given in Appendix V. The following observations have been made after the computational experiments. The proposed algorithm converges quickly to the good solutions for small problems (i.e. it takes maximum 3 minutes for problems with 12 machines, 15 parts and 10 REs). For medium size problems it requires around 8 minutes to produce a good quality solution (i.e. 20 machines, 25 parts and 12-15 REs). For bigger size problems it requires more than 15 minutes to converge a good quality solution. The selection of Tabu search parameters may also effect the quality of the solutions. However, an extensive experimental study is required to access the computational time performance in relation to quality of the generated solutions. Quality of the generated solution can be assessed by determining the lower bounds for a given problem.

6.5 CONCLUSIONS

A capability-based approach to solve manufacturing cell formation problems is proposed in this chapter. The model is formally stated as a pre-emptive goal programming problem. Minimising dissimilarity between parts in each cell, minimising total load inbalance between cells, minimising extra capacity requirements while configuring the cellular shop and maximising cell flexibility are considered as the objectives to be optimised. Maximum and minimum number of parts and machines in each cell and cell independence are considered as the constraints. The developed mathematical model is solved with the multiple objective TS algorithm that was developed in Chapter 5. Some problem-specific additions to the original TS algorithm are also discussed and presented. The proposed model simultaneously solves cell formation problems. Although the proposed model is sophisticated, its size is noticeably smaller than those of some other known models (e.g. Kusiak's *p-median* model). A new approach is also proposed for capacity feasibility analysis that reduces the detrimental effects of non-integer machine requirements (Sule, 1991).

As the results of comparative work show, the capability based cell formation technique used in MOCACEF 1.0 gives better results. The manufacturing cells are formed with improved capacity utilisation levels and reduced extra machine requirements. It is also more likely to design independent manufacturing cells with higher flexibility by using the RE concept in cell formation. These factors are also known as the main shortcomings of CMSs.

In the present approach used in MOCACEF 1.0 overlapping capabilities between machines, and alternative machines for part processing can be automatically taken into account while forming cells. This is the main reason why the present cell formation model produced better results. Therefore it can be concluded that the capabilities of production resources must be realised while solving cell formation problems. This issue is especially important when the existing job shop contains many highly automated machining centres. These machines generally have a considerable amount of overlapping capability. Hence, it is not advantageous to assign a fixed machine route for components (if not necessary for other reasons) which eliminates the opportunity of utilising the alternative resources and flexibility available in the job shop. However it is also needs to be stated that consideration of all alternative machines for part processing may increase component movement between machines and therefore may increase the material handling cost.

A Computer program MOCACEF 1.0 has been developed during this study as part of a decision support-framework to solve cell formation problems. A stand-alone version of MOCACEF 1.0 in FORTRAN-90 is given in Appendix IV.

CHAPTER SEVEN

7. DEVELOPMENT OF MULTIPLE OBJECTIVE CELL LOADING MODULE

7.1 INTRODUCTION

In dynamic manufacturing environments, part families constantly change due to changes in design, demand and introduction of new part types. Under these circumstances the loading (or part assignment) problem should be considered seriously in order to retain the performance of CMS. As was discussed in Chapter 2, Greene and Sadowski (1980,1983) classified loading as a controlling activity with scheduling under the general topic of "controlling CMS". Loading was responsible for distributing parts among feasible cells and scheduling was responsible for controlling parts inside cells (i.e. distributing parts among machines in each cell and their timing). Due to confusion between the terms loading and scheduling in job-shop applications, the latter one is considered to contain the former one or vice-versa in some other environments. In this Chapter, loading is considered as the main controlling activity in CMS and it is defined as follows: *loading in CMS is the determination of how a part should be assigned to a cell (or cells) in the current production period to meet the required performance levels.* The determination of 'how to assign' can be named as the *part assignment problem* and the determination of the performance is related to the operation of system via activities such as

scheduling. Therefore, the loading problem can be considered as the combination of *part to cell assignment* and *cell scheduling* problems in CMS (see Figure 7.1).

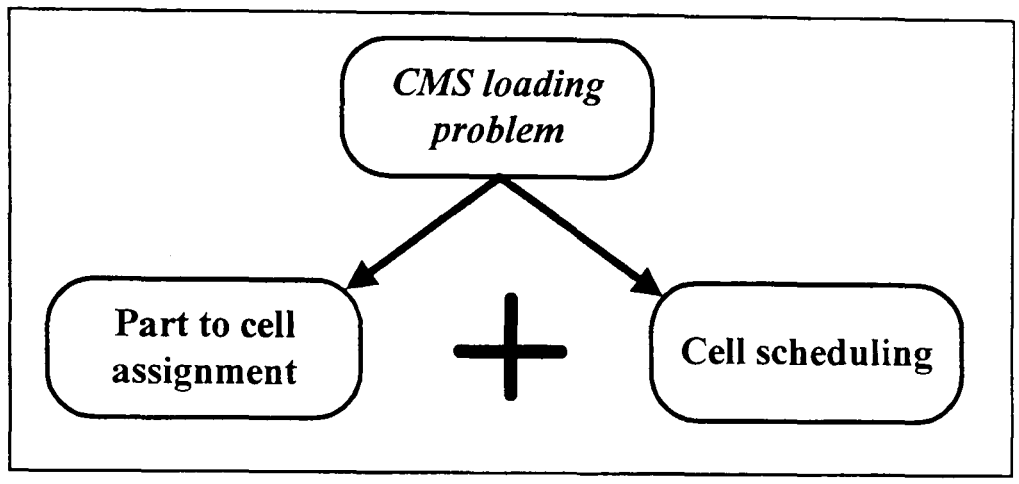


Figure 7.1 Components of loading problem in CMS

Loading in CMS is a complex problem. It requires simultaneous solution of the *part to cell assignment* and *cell scheduling* problems by considering specific system constraints and multiple objectives. In CMS applications, manufacturing cells are considered as independent units inside the factory. The minimisation of interaction between cells is necessary to have a true CMS and its potential advantages (Kusiak, 1990). Increasing the interaction between cells increases the complexity of the system and many advantages of CMS disappear. Therefore, the CMS loading problem should be considered from this perspective, loading alternatives that result in minimum component movements between cells should be determined, and the one that closely meets the performance requirements should be found. The overall efficiency of CMSs facing changing production requirements can be improved by such a loading strategy.

In this chapter, a loading approach for CMSs is proposed. As briefly explained in Chapter 3, the proposed approach is composed of two integrated sub-modules. In the first module the *Multiple Objective Tabu Search* is used to generate the loading scenarios then, in the second module the *Simulation based Scheduling System* is used to obtain the performance of the generated loading scenarios and the final production schedule respectively. The *parts list* (see Chapter 3) that contains the RE-based process plans is the main input of the proposed loading system. The process plans in the *parts list* are machine-independent abstract process plans. The final machine-based process plans are the outcome of the loading system.

In the following sections of this chapter, the loading problems in cellular manufacturing applications are defined, refinements to the original multiple objective Tabu search algorithm are explained, the simulation and scheduling module is presented, and an example application is illustrated.

7.2 PROBLEM STATEMENT

The loading problem in CMS can be represented in a schematic form, as shown in Figure 7.2. The *parts list* contains a set of jobs that must be assigned to the existing manufacturing cells for processing in the current production period. The processing requirements of each part type are defined in terms of REs. The machining capability of each machine tool in each cell is also represented by a set of REs. The overall capability of each cell is the union of capabilities of its resources as explained in Appendix III. Each part is represented by a string that contains necessary information in terms of RE for processing (i.e. operation numbers, RE requirements, machining

times etc.). Each cell is also represented by a string that contains information about its processing capability in terms of REs (see Figure 7.2).

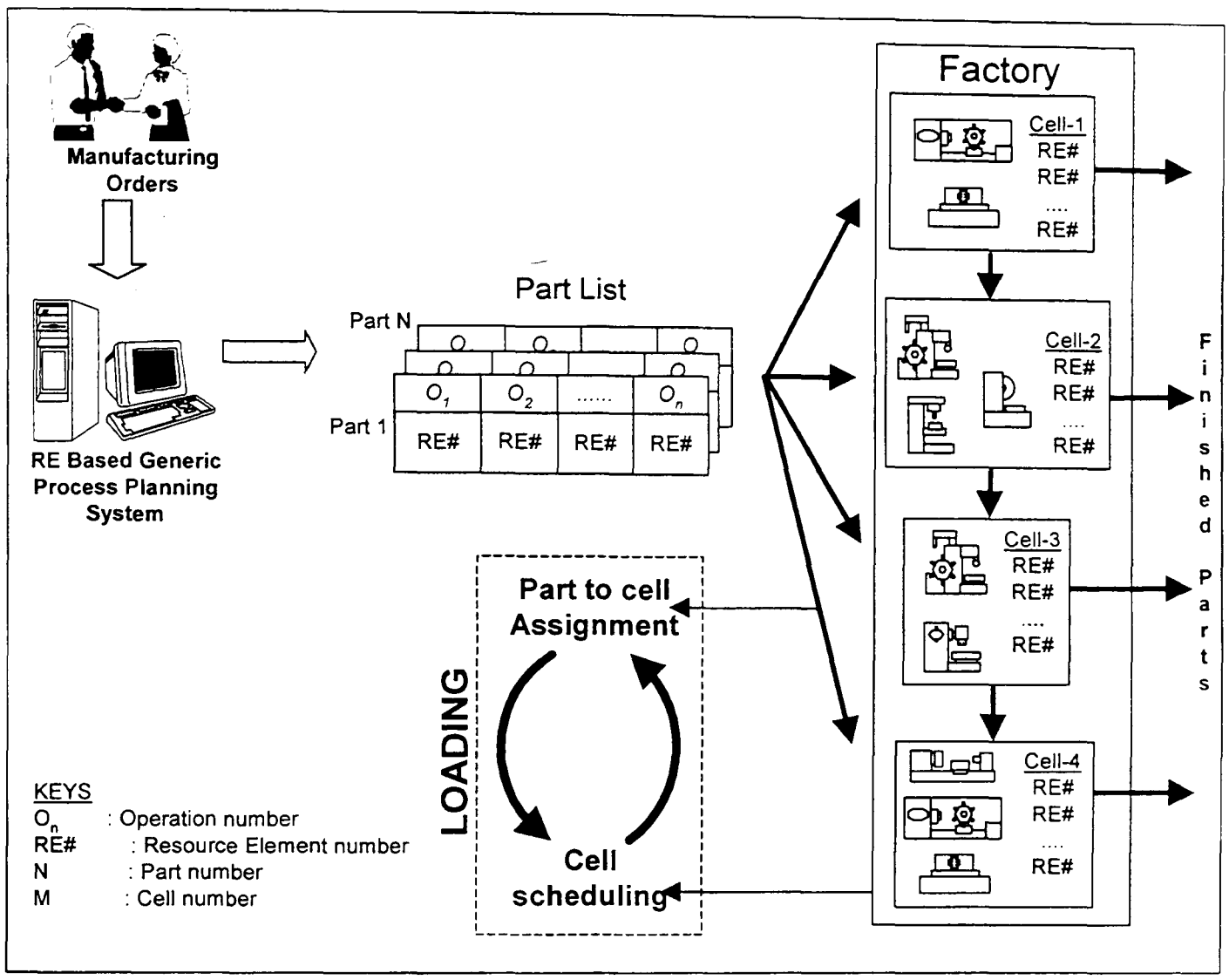


Figure 7.2 Graphical description of the CMS loading problem

It may not be possible to assign all operations of a part type to only one cell because such a target cell that can satisfy all processing requirements may not be available. Inter-cell movement must be considered in such cases. The *set of cells* for each part type is defined as the *alternative cells assignment* that results minimum inter-cell movement. An example network to determine these alternatives is shown in Figure 7.3.

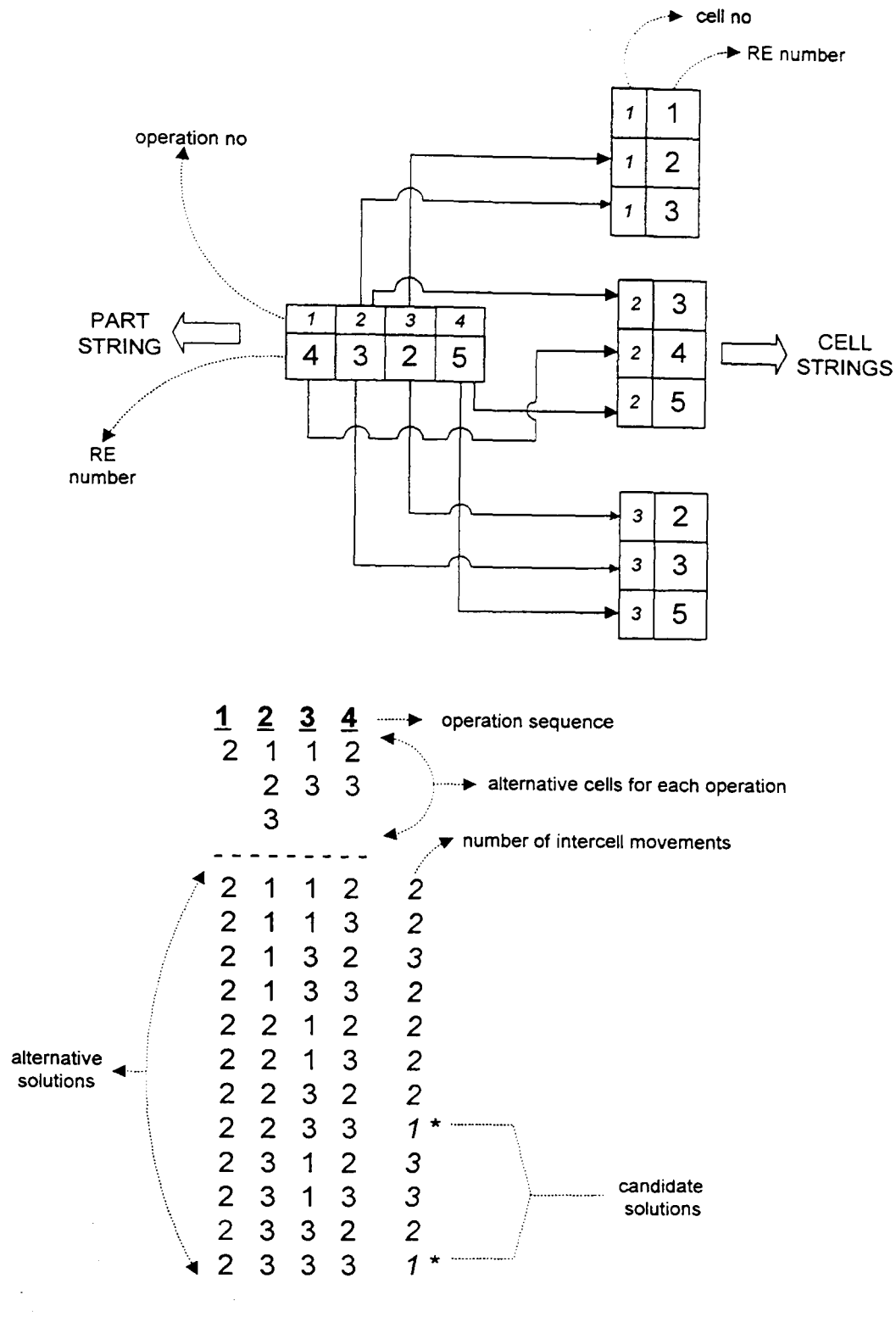


Figure 7.3 An example string for a part type and determination of its alternative cells assignment (machining time data is not shown in the string)

The set of *alternative cells assignments* for all parts in the *part list* constitute a directed graph (MIG: Minimum interaction graph (Baykasoglu, *et. al.*, 1998-a). MIG represents all possible part-to-cell assignment alternatives that results minimum cell interaction in the CMS (a hypothetical example MIG graph is shown in Figure 7.4)

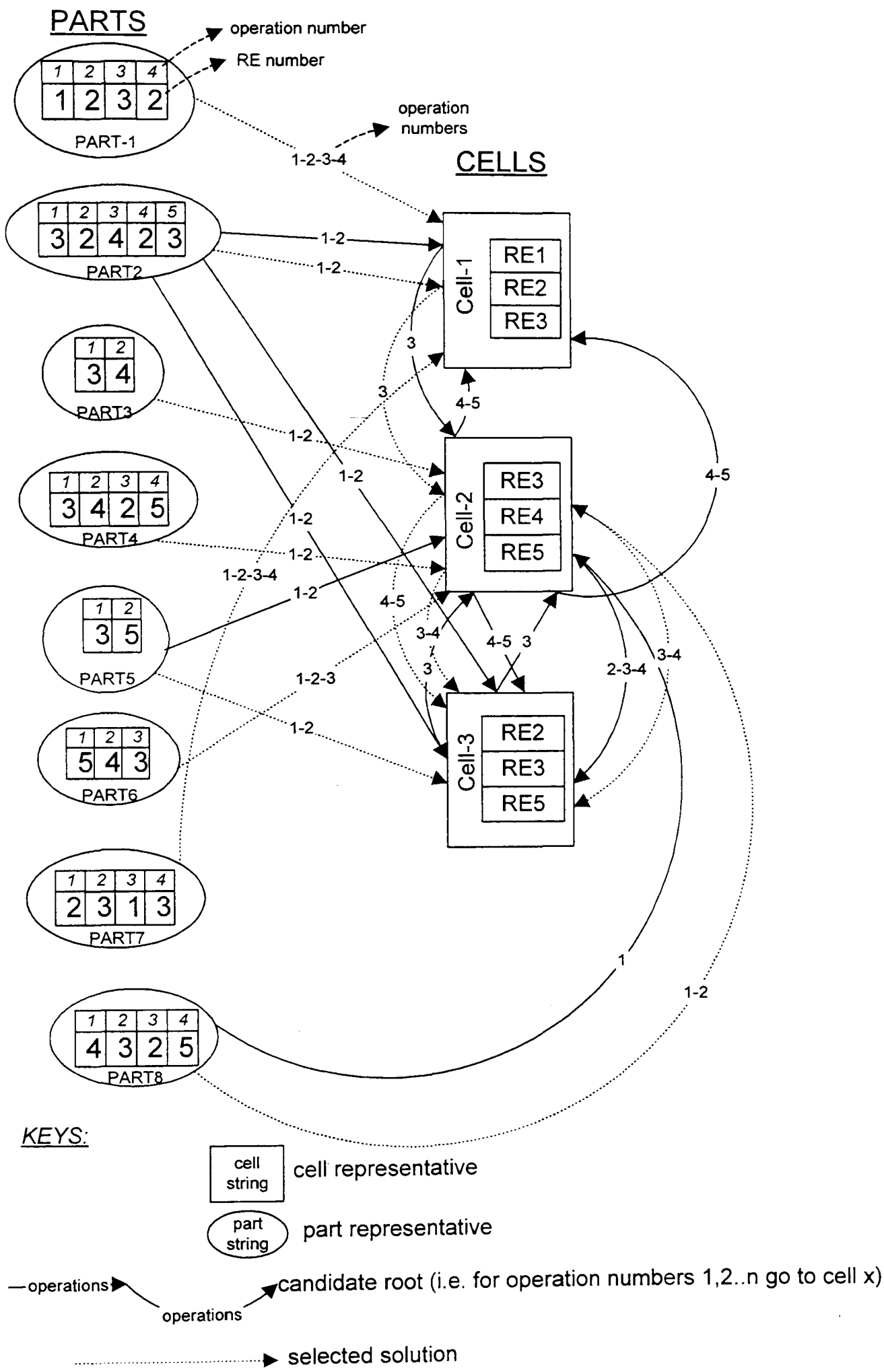


Figure 7.4 Construction of the MIG graph for a hypothetical CMS with eight part types and three manufacturing cells

In Figure 7.4 Part-1 is assigned to Cell-1 for all of its operations (i.e. operations 1,2,3,4 as shown on the arrow) that correspond RE1, RE2, RE3, RE2 because Cell-1 can supply all these REs. There is no other cell that can satisfy all processing requirements of Part-1, all other assignments result in inter-cell movement, therefore only this assignment is included in the MIG graph.

A solution amongst the alternative solutions available on the MIG graph that best satisfies the multiple performance requirements (i.e. mean tardiness, overall utilisation and total throughput) should be found. If it is not possible to find such a solution then system reconfiguration should be considered. This kind of decision-making problem may frequently occur in a CMS facing changing production requirements.

The size of the solution space is directly related to the problem size. If the problem size gets bigger the solution space increases enormously. In such cases *complete enumeration* of all possible solutions on the MIG graph may not be computationally feasible. The total number of part assignment scenarios, LS (i.e. size of the part assignment problem) can be determined by the following equation:

$$LS = \prod_{i=1}^{N} RT_i \qquad 7.1$$

where N is the total number of parts and RT_i is the number of alternative routes for part i (see Figure 7.3)). For a problem with 20 parts and each part having 3 *alternative cell assignments* there are 3486784399 possible part assignment scenarios. It is clear that for the solution of large problems, a *complete enumeration*

strategy is not feasible. Therefore, effective heuristic procedures are required. The Tabu search algorithm (see Chapter 5) is employed to solve CMS loading problems.

From its nature, the problem is a multiple objective type and there are priorities between certain types of objectives. For example in this study cell interaction is considered as the most important objective since inter-cell movements in a CMS should be minimised as much as possible in order to obtain the true CMS with most of its advantages. The CMS loading problem is formally represented as a pre-emptive goal programming model. The multiple-objective Tabu search algorithm that was introduced in Chapter 5 is used to solve the mathematical model. The problem specific modifications to the original TS algorithm are explained in detail in the following sections. At a lower level, within a cell, (see Figure 7.2), during the assignment of parts to machines (the production schedule), practitioners may want to optimise more than one objective at the same time. A simulation based scheduling approach is employed to generate and simulate the production schedule dynamically. Therefore, the objective is to introduce an integrated system by which the part assignment and scheduling scenario (i.e. loading) to be finalised is not only based upon a high level of decision-making, but also upon the operational level within cells. Consequently, the generated solution is unlikely to be inapplicable.

7.3 MODELLING AND SOLUTION APPROACH

The loading problem is represented in a pre-emptive goal programming framework with the following performance measures: *cell interaction, mean tardiness in the system*, *system utilisation* and *total throughput from the system*. The first objective function value is obtained from an analytically defined equation and the last three are

obtained from the simulation and scheduling module. The objective is to solve the loading problem in an integrated environment and to make a decision on the final part assignment and scheduling scenario based upon the system performance measures obtained from the simulation. The multiple-objective tabu search based algorithm is used for solving the simulation optimisation model. The simulation and scheduling module (Gindy and Saad, 1996,1997,1998) is developed by using the SIMAN V (Pegden, *et. al.,* 1990) simulation language and C/C++ general-purpose computer programming language. A simplified flowchart of the proposed integrated system is presented in Figure 7.5. The following sub-sections explain in detail the entities of the proposed system.

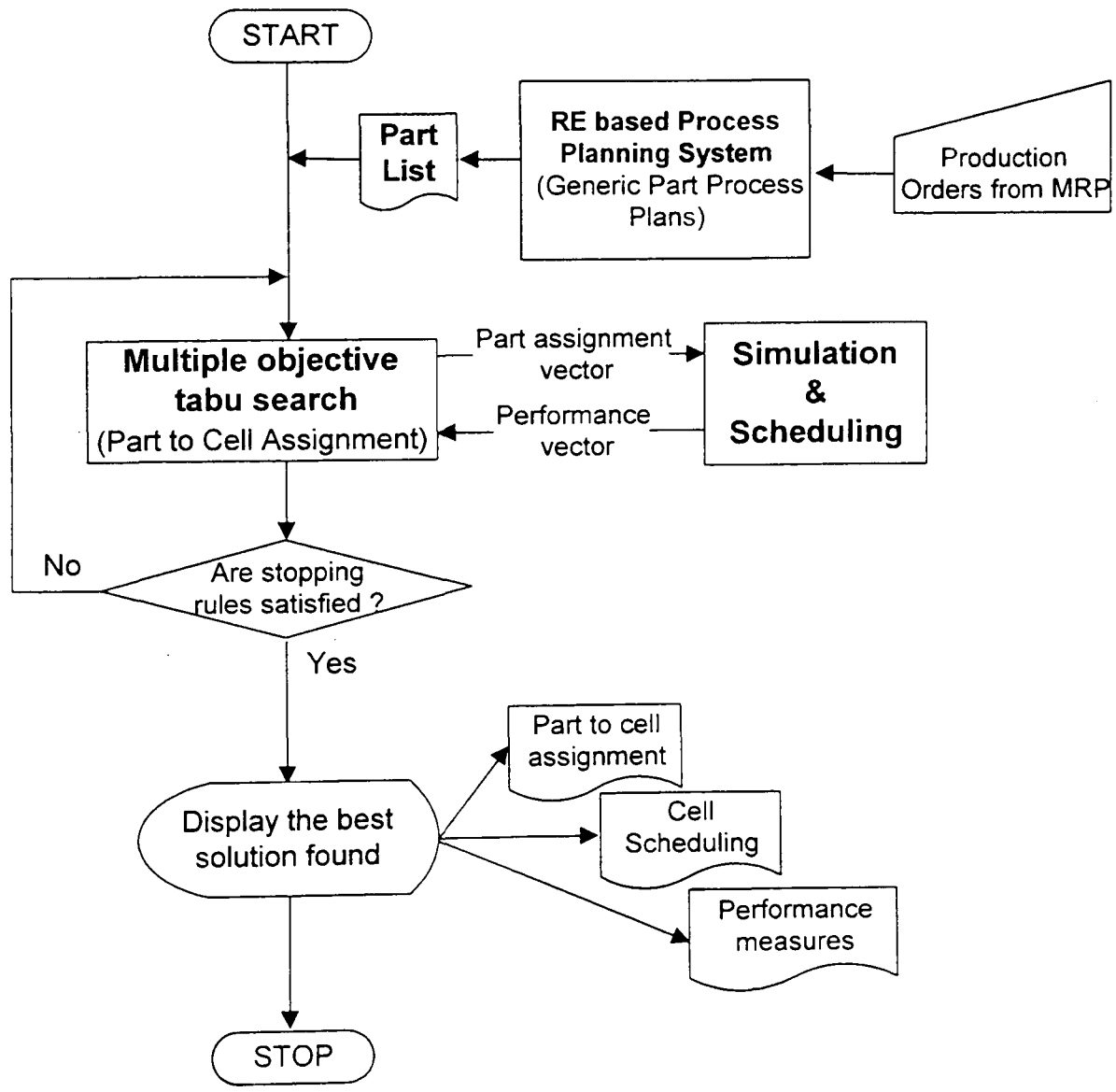


Figure 7.5 A simplified flowchart of the loading system

7.3.1 Mathematical Modelling

The following notation is used in the mathematical model:

Indexes

i: Part index

o: Operation index

k: Cell index

Parameters

MC: Number of cells in the current production period

NOP_i: Number of operations in part i (each operation corresponds to a RE)

R_{io}: RE type (defined by a number) corresponding the o'th operation of part i

$C_{(Rio)k}$: If RE corresponding the o'th operation of part i is available in cell k 1, otherwise 0

$\boldsymbol{MT}$: Mean tardiness in the CM system (determined by the simulation module)

$\boldsymbol{MU}$: Mean utilisation of the CM system (determined by the simulation module)

$\boldsymbol{TT}$: Total throughput in the CM system (determined by the simulation module)

goal_1: Acceptable level of inter-cell movement

goal_2: Acceptable limit for tardiness

goal_3: Desired level of system utilisation

goal_4: Desired level of throughput

Variables

X_{iok}: 1 if o'th operation (corresponding RE) of part i is assigned to cell k, 0 otherwise

d_c^-, d_c^+: Under and over achievement of *goal_1*

d_t^-, d_t^+: Under and over achievement of *goal_2*

d_u^-, d_u^+: Under and over achievement of *goal_3*

d_q^-, d_q^+: Under and over achievement of *goal_4*

The pre-emptive goal programming model is given as follows;

$$\text{lexmin}\{(d_c^+),(d_t^- + d_t^+),(d_u^- + d_u^+),(d_q^- + d_q^+)\} \tag{7.2}$$

$$\sum_{i=1}^{m}\sum_{o=1}^{NOP_i-1}\sum_{k=1}^{MC}\left|X_{iok} - X_{i(o+1)k}\right| + d_c^- - d_c^+ = goal\_1 \tag{7.3}$$

$$\boldsymbol{MT} + d_t^- - d_t^+ = goal\_2 \tag{7.4}$$

$$\boldsymbol{MU} + d_u^- - d_u^+ = goal\_3 \tag{7.5}$$

$$\boldsymbol{TT} + d_q^- - d_q^+ = goal\_4 \tag{7.6}$$

$$\sum_{k=1}^{MC} X_{iok} = 1 \quad \forall i,o \tag{7.7}$$

$$\sum_{k=1}^{MC} C_{(R_{io})k} - X_{iok} \geq 0 \quad \forall i,o \tag{7.8}$$

$$X_{iok} \in \{0,1\} \quad \& \quad d_c^-, d_c^+, d_t^-, d_t^+, d_u^-, d_u^+, d_q^-, d_q^+ \geq 0 \quad \forall i,k,o \tag{7.9}$$

Equation 7.2 represents the lexicographical order of deviational variables to be minimised. Equation 7.3 represents the first goal constraint (cell interaction). For a part assignment scenario this equation calculates the total number of inter-cell movements. Equations 7.4, 7.5 and 7.6 are second, third and fourth goal constraints respectively. For a part assignment scenario the values of performance indicators, ***MT***, ***MU*** and ***TT*** are determined from the simulation and scheduling module which is explained in subsection 7.3.3. Equation 7.7 ensures that every operation of a part are assigned to only one cell. Equation 7.8 ensures that, if an operation of a part is assigned to a cell the RE requirement corresponding to that operation is available in that cell.

7.3.2 Application of Multiple Objective Tabu Search Algorithm to Solve Loading Model

The multiple-objective TS algorithm that was developed in Chapter 5 is employed to solve the loading model. However, a number of problem specific features are incorporated into the original TS algorithm. The original TS algorithm can also be used without any change. The following problem-related modifications are made to speed up and improve the performance of the original multiple objective TS algorithm:

Generation of neighbourhood solutions:

Instead of using neighbourhood generation functions that were presented in Chapter 4-Section 4.2.1., the following movement strategy is applied for generating the neighbourhood solutions from the current solution (the current solution is the initial solution in the first iteration of the TS, in the subsequent steps it is the best neighbourhood solution):

- *Step-1:* Select a part randomly from the current solution.
- *Step-2:* Select a cell randomly.
- *Step-3:* Starting from the first operation in the processing sequence of the randomly selected part assign it to the randomly selected cell if it can supply the required RE by this operation, if not select another cell randomly which can supply it. Continue with the next operation in the processing sequence, if the RE required by this operation can be supplied by the current cell assign it to this cell otherwise choose another cell randomly which can supply it. Continue to this process until all operations of the part are assigned.
- *Step-4:* Iterate Steps 1 to 3 until required number of neighbourhood solutions are generated from the current solution.

By employing this strategy one can easily generate feasible neighbourhood solutions that satisfy all hard constraints (Equations 7.7, 7.8). An illustrative example is shown in Figure 7.6, in which there are three parts (1st part has 3 operations, 2nd part has 3 operations and 3th part has 4 operations), three cells and neighbourhood size is 2.

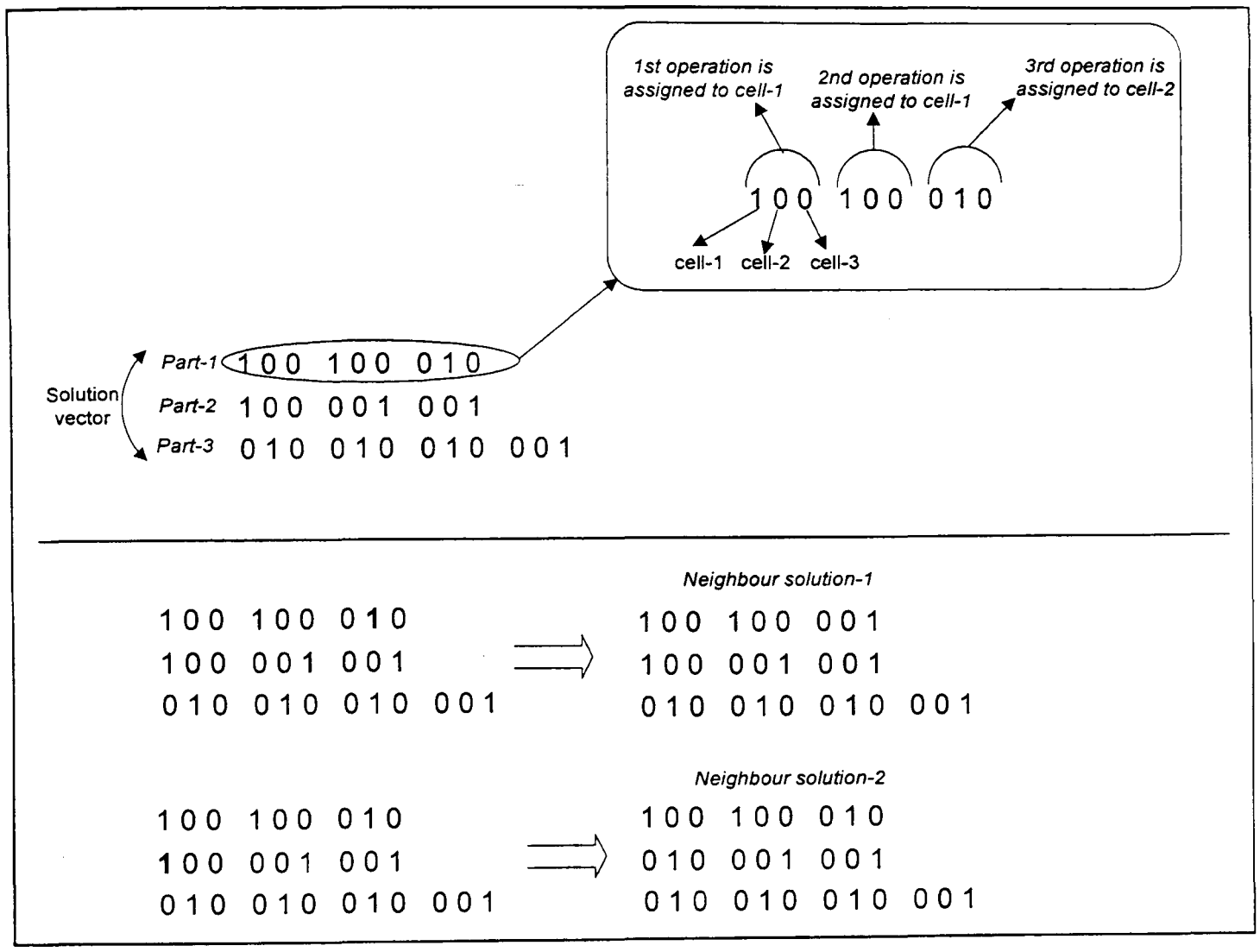


Figure 7.6 Neighbour solutions generation (randomly selected parts are bolded)

Tabu list:

Indices of randomly selected and reassigned parts are put into the *tabu list*.

Aspiration criteria:

Any move which improves the best known solution is accepted, even if the move is tabu (see also Chapter 6-Section 6.3 for more details about aspiration criteria).

7.3.3 Simulation and Scheduling Module (SSM)

In general, scheduling in a manufacturing system is the process of assigning the starting and completion times to jobs. Managers generally want to process these jobs as efficiently as possible, optimising perhaps flow time in the system, tardiness, utilisation of equipment or some other objective. The problem of satisfying more than one objective is formulated as a multiple objective optimisation problem in which the goal is to minimise or maximise not a single objective function but several functions simultaneously. As mentioned in section 7.3.1 above, four different objective functions are considered in the proposed integrated system. Three of them (mean tardiness, system utilisation, and throughput) are obtained from the simulation and scheduling module (SSM).

In the SSM, the simulation model of a manufacturing facility that can be operated as a collection of Resource Elements has been built by Saad (Gindy and Saad, 1997). Dispatching rules are used as the basis for generating the production schedule. The objective of creating dynamic scheduling is to be adaptive and able to respond quickly and effectively to changes. The basic idea of the scheduling system is the use of generic (machine independent) process plans associated with each part and leaving the decision of matching parts to machines according to the processing requirements of the parts (in terms of Resource Elements) and the processing capabilities of the machines (also in terms of Resource Elements) to a later stage, to improve performance levels. Then the final part assignment scenario and the production schedule are generated simultaneously.

The simulation and scheduling module provides a wide range of dispatching and due-date assignment approaches to schedule the production and also to help the manager to better set due-dates for the jobs. The comparison between these rules is beyond the scope of this study, however a detailed comparison has been done by the other members of the Responsive Manufacturing Centre (Gindy and Saad, 1996, 1997, 1998). In this research the *Earliest Due-Date* rule is used as a dispatching rule (i.e. select the job which has the earliest due-date) and the *Total Work Content* rule as a due-date assignment approach (i.e. set the job's allowance equal to the sum of its total processing time plus an allowance proportion to the total work content). However, any other dispatching and due-date assignment rules can be used if required. It is also worth to noting that the research done by Gindy and Saad (1996, 1997) has shown that RE-based scheduling is less sensitive to the selection of these rules. It was also shown that the RE-based scheduling strategy significantly outperforms the traditional machine-based scheduling strategies. In Tables 7.1 and 7.2 a set of most commonly used dispatching and due date assignment rules are given (Blackstone, 1982, Brah, 1996).

Table 7.1 Dispatching rules

Name of the dispatching rule	*Symbol*	*Definition*
Earliest Due-Date	***EDD***	Select the job which has the earliest due date: $JOB_{ij} = DD_i$
Minimum Slack Time	*MST*	Gives priority to the minimum slack time: $JOB_{ij} = DD_i - t + (TPT_i - \sum_{t=0}^{t} OP_{ij})$
Smallest Remaining Slack per Operation	*SOPN*	Gives priority to the minimum remaining slack per operation: $JOB_{ij} = DD_i\, t / (O_i - o_i(t))$
Shortest Processing Time	*SPT*	Select the job that has smallest operation time: $JOB_{ij} = OP_{ij}$

Table 7.2 Due-date assignment rules

Name of the due date assignment rule	*Symbol*	*Definition*
Total Work Content	***TWK***	Set job's allowance is equal to the sum of its total processing time plus a constant an allowance proportion to the total work content: $TWK_i = \alpha TPT_i$
Processing Plus Waiting	*PPW*	Set job's allowance is equal to the sum of its total processing time plus an allowance for non-productive inter-operation activities (i.e. waiting in queues etc.): $PPW_i = TPT_i + \bar{TPT}(\alpha - 1)O_i / \bar{O}$
Constant Slack Approach	*SLK*	Set job's allowance is equal to the sum of its total processing time plus a constant slack calculated based on the mean processing time of all jobs in the system: $SLK_i = TPT_i + \bar{TPT}(\alpha - 1)$

In Tables 7.1 and 7.2, i is the job index, j is the operation index, DD_i is the due date of job i, OP_{ij} is the processing time required for the jth operation of job i, JOB_{ij} is the priority index of the ith job at the jth operation, t is the present time, TPT_i is the total processing time of job i, O_i is total number of operations of job i, $o_i(t)$ is total number of operations of job i already done at time t, α is the due-date tightness, $\bar{TPT}$ is the mean processing time and $\bar{O}$ is the mean number of operations.

Figure 7.7 displays the simulation and scheduling module (SSM). As can be seen, the inputs to the SSM are the part assignment scenario and the process plans for each job, and the outputs are the production schedule and the performance indicators. The SSM includes the presentation of the existing resources in a system, scheduling engine in terms of dispatching rules, due-date assignment approach and the operation and control policies of the system.

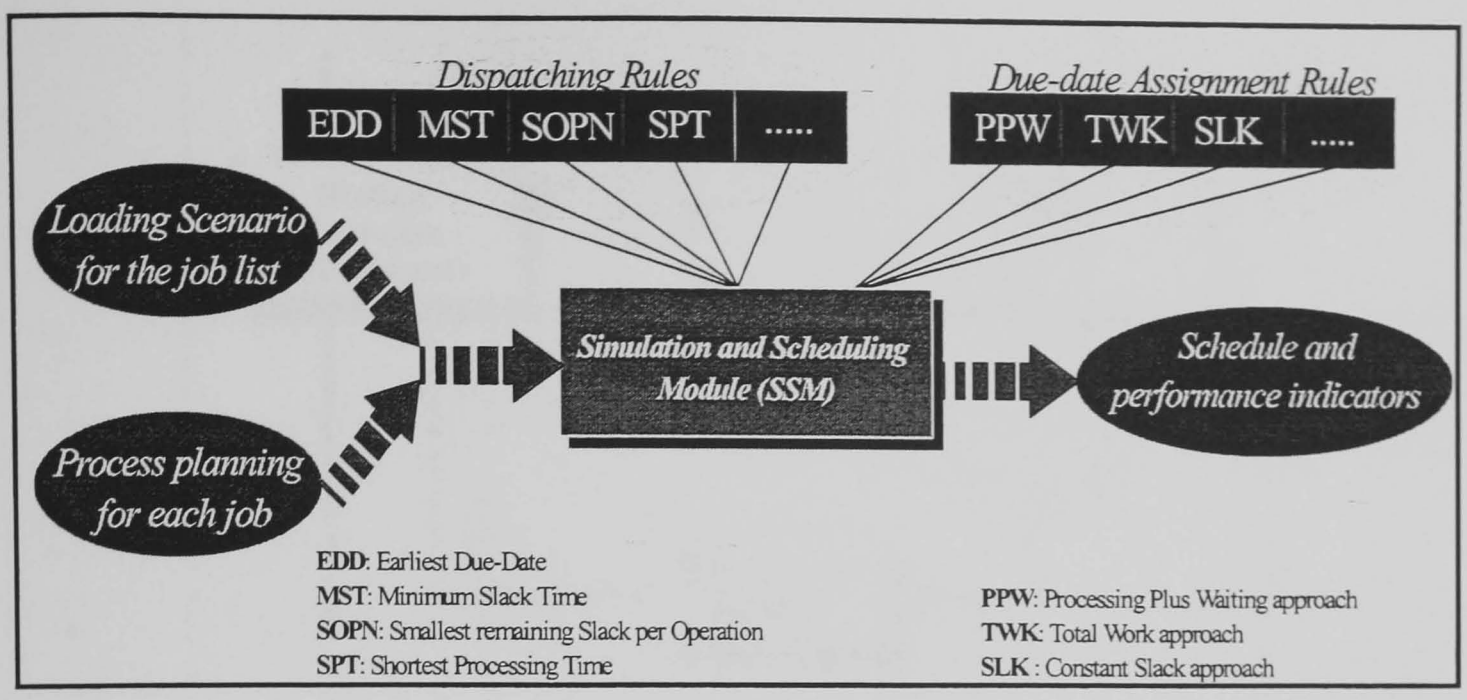


Figure 7.7 Simulation and scheduling module

7.3.4 Generation of the Loading and Scheduling Scenario (The simulation optimisation cycle)

The proposed integrated system starts with an *initial solution* that is randomly generated using the Tabu search algorithm or starts with a known feasible solution. A set of neighbourhood solutions which represent the possible *part assignment scenarios* are generated by the Tabu search algorithm. Then these solutions are passed to the simulation and scheduling module for evaluation. Each time a *part assignment scenario* is generated the simulation model is automatically modified (via the parametric simulation approach[1]) for this scenario by a specially developed C/C++ subroutine to represent the new situation. Figure 7.8 graphically depicts the parametric simulation system.

[1] In parametric simulation approach a part (or whole) of the simulation program is dynamically regenerated (i.e. the model itself is a parameter) in relation to new requirements (i.e. new neighbourhood solutions where new feasible part to cell assignments are generated) see also Chapter 2-Section 2.4.3.

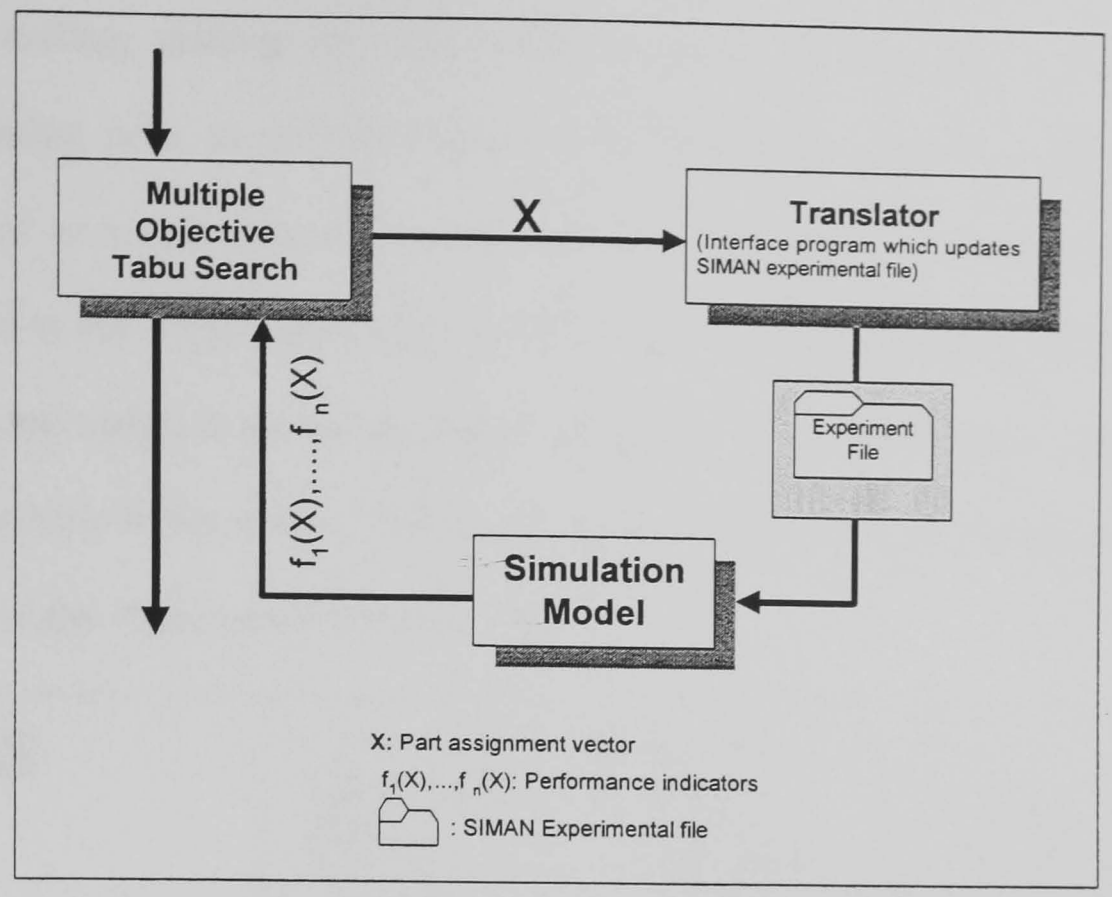


Figure 7.8 Parametric simulation system

The optimisation cycle starts once the simulation and scheduling module sends back the performance vector to the Tabu search algorithm (refer Figures 7.5 and 7.8), at this stage the Tabu search algorithm calculates deviational variables, selects the best current solution, updates the best known solution and generates new neighbourhood solutions. This process continues until convergence. After convergence, the system will stop and generate two different outputs: first, the part assignment scenario with the performance indicators associated with this particular scenario, second, the detailed production schedule.

7.4 EXPERIMENTAL WORK

The proposed system has been applied to a hypothetical manufacturing facility containing 12 machine tools that perform a wide variety of machining operations e.g.

turning, milling, drilling etc. The machine tools are assigned to four virtual manufacturing cells as an initial solution (the previous production periods) and represented as a collection of resource elements (see Table 7.3) and the parts are dispatched to the system according to their *RE*-based generic routes (see Table 7.4). The parts and materials are transported in the system by an AGV system. There are 5 AGVs available in the system. The layout of the system is shown in Figure 7.9. The AGV net is also represented in Figure 7.9.

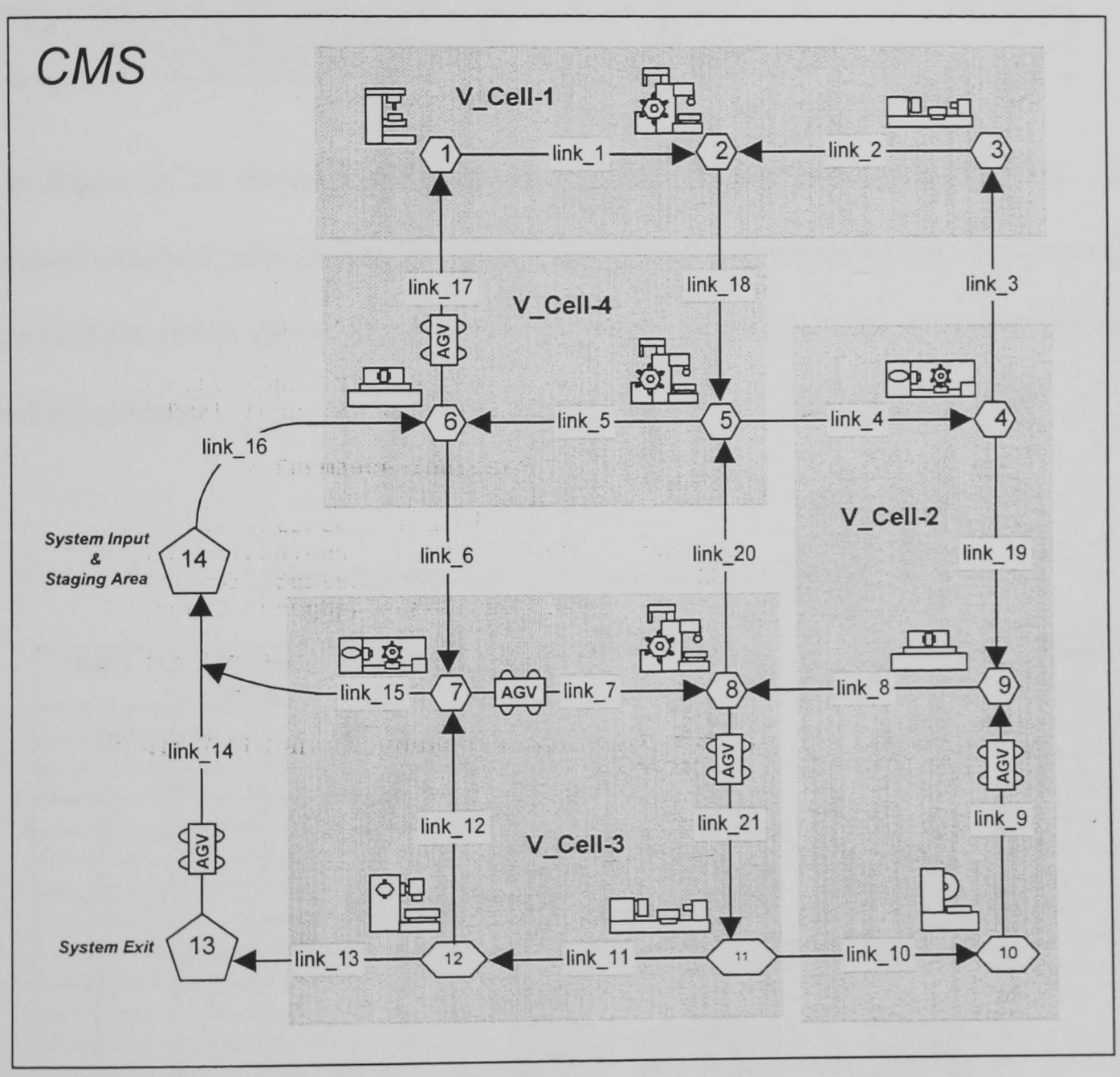


Figure 7.9 Layout of the prototype CM system

Table 7.3 Virtual cell capabilities based on *REs*

Virtual Cells	*Machine Tools*	*Resource Elements*										
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
1	1-Drill Press-1	*										
	2-MHP Machining Centre-1	*	*	*				*	*	*	*	*
	3-Colchester Lathe-1	*	*		*			*				
2	4-MHP MT50 NC Lathe-2	*	*		*			*			*	
	9-CNC Grinding Machine-2					*	*					
	10-J&S Cyc. Grinder						*					
3	7-MHP MT50 NC Lathe-1	*	*		*			*			*	
	8-MHP Machining Centre-2	*	*	*				*	*	*	*	*
	11-Colchester Lathe-2	*	*		*			*				
	12- J&S Surf. Grinder					*						
4	5-MHP Machining Centre-3	*	*	*				*	*	*	*	*
	6-CNC Grinding Machine-1					*	*					

Production of 20 different part types is required. Part processing requirements and related technical information as generic process plans in terms of REs are listed in the *part list* that is shown in Table 7.4. All values in the table are assumed values to test the system.

Table 7.4 Part list: generic part process plans

Part #	# Operations (NOP_i)	# RE	Operation # 1	Operation # 2	Operation # 3	Operation # 4
1	3	3	RE1(60)*	RE2(80)	RE4(90)	
2	3	3	RE1(50)	RE2(60)	RE3(40)	
3	3	3	RE5(20)	RE6(60)	RE7(80)	
4	2	2	RE8(40)	RE5(50)		
5	3	3	RE7(50)	RE4(60)	RE5(80)	
6	3	3	RE8(50)	RE6(50)	RE7(60)	
7	3	3	RE8(70)	RE9(70)	RE10(80)	
8	3	3	RE9(60)	RE10(50)	RE11(90)	
9	3	3	RE5(70)	RE1(50)	RE2(30)	
10	2	2	RE3(40)	RE4(50)		
11	3	3	RE5(50)	RE6(50)	RE9(40)	
12	3	3	RE10(70)	RE8(80)	RE9(80)	
13	3	3	RE5(10)	RE8(30)	RE10(50)	
14	3	3	RE8(50)	RE7(40)	RE5(50)	
15	2	2	RE1(60)	RE2(20)		
16	2	2	RE3(60)	RE4(30)		
17	3	3	RE6(40)	RE7(50)	RE8(70)	
18	4	4	RE8(50)	RE9(50)	RE10(50)	RE11(20)
19	2	2	RE5(50)	RE2(50)		
20	3	3	RE7(10)	RE8(50)	RE9(30)	

* RE#(A): Resource Element number(Processing time + set up time)

The proposed integrated system (*the loading module*) was applied to the above case. The inter-cell movement limit (i.e. goal) was set to zero, the tardiness goal was set to zero, the utilisation goal was set to %65, and the total throughput goal was set to 5000 parts. The number of iterations was 250 and in each iteration three neighbourhood solutions were generated. This required 750 simulation experiments. The following operating assumptions are made in the simulation:

- Deterministic processing time at each machine tool.
- Set up time is included in the operation time.
- Production orders arrive at the system using exponential distribution for the demand in the planning period.
- Machine breakdown and corrective maintenance work do not occur during the production run.
- Due dates are calculated using the TWK approach as explained in section 7.3.3.
- Dispatcher uses EDD rule as explained in section 7.3.3.

The proposed integrated system was developed and executed on a PC 32 MB RAM and 200 MHz. The time required to finish the 250^{th} iteration was 123 minutes. This means that the proposed system requires modest computing power to generate the solution. The time will vary depending upon the computing power available and the size of the problem (i.e. the number of neighbourhood solutions and the number of iterations).

The results obtained from the experimental study are shown in Figures 7.10-7.12. The interpretation of results shows that inter-cell movement goal has nearly been satisfied (i.e. only one part type requires inter-cell movement). This may be due to

the existing overlapping capabilities between machine tools and cells that give flexibility in loading the entire system. Although the system converged to a good solution, the satisfaction level of objectives may not be acceptable to the decision-maker. In this case one possible action that can be taken is the modification of the existing configuration to improve the solution (reconfiguration action). Reconfiguration is explained in the following chapter by continuing the same example. Figure 7.10 shows the conversion behaviour of the performance measures considered in this research work. Figure 7.10-a displays the convergence of inter-cell movement performance. Figure 7.10-b displays the convergence of the mean tardiness performance and so forth. As more important measures are improving the less important measures may not improve or may temporarily deteriorate. There is a deterioration between iteration numbers 3-18 in the mean tardiness performance because in the same interval the inter-cell movement objective is improving which was considered as the most important objective. The same behaviour can also be observed for the other performance measures. Nevertheless, when a higher importance measure stabilises the lower importance measures continue to improve to better values as can be seen from convergence graphics in Figure 7.10. Consequently all objectives are tried to be optimised simultaneously.

Typical part-to-cell assignment and production schedule information for the best solution generated from the developed integrated loading system are presented in Figures 7.11 and 7.12 respectively. Figure 7.11 depicts the part to cell assignment output that consists of information about the route of each part to be processed in the system and some performance indicators associated with this particular part-to-cell assignment scenario. For example, part type number 1 needs three different

operations, first is Resource Element number 1 which will be provided from cell number 1, second, is Resource Element number 2, the part is going to get this resource from cell 1 and the third, is Resource Element number 4 which is going to be provided by cell 1 as well, and so forth for the rest of the twenty part types in the list. The performance indicators that have been used to make a decision on the final loading are shown at the bottom of Figure 7.11, they are: inter-cell load transfer, mean tardiness, overall system utilisation and the total throughput from the system. Figure 7.12 depicts a part of the generated production schedule. As can be seen from Figure 7.12, scheduling output consists of information about the machine route of each part in each cell and their start and finish times. For example, Part Type-12 (which is assigned to cell-1, see Figure 7.11) with ID number 2 is processed on Machine-2 for its first operation (which corresponds to RE10) between 0-70 sec., for its second operation (RE8) is processed on Machine-2 between 70-150 sec., for its last operation (RE9) is again processed on Machine-2 between 151-231 sec. and so forth for the rest of the parts.

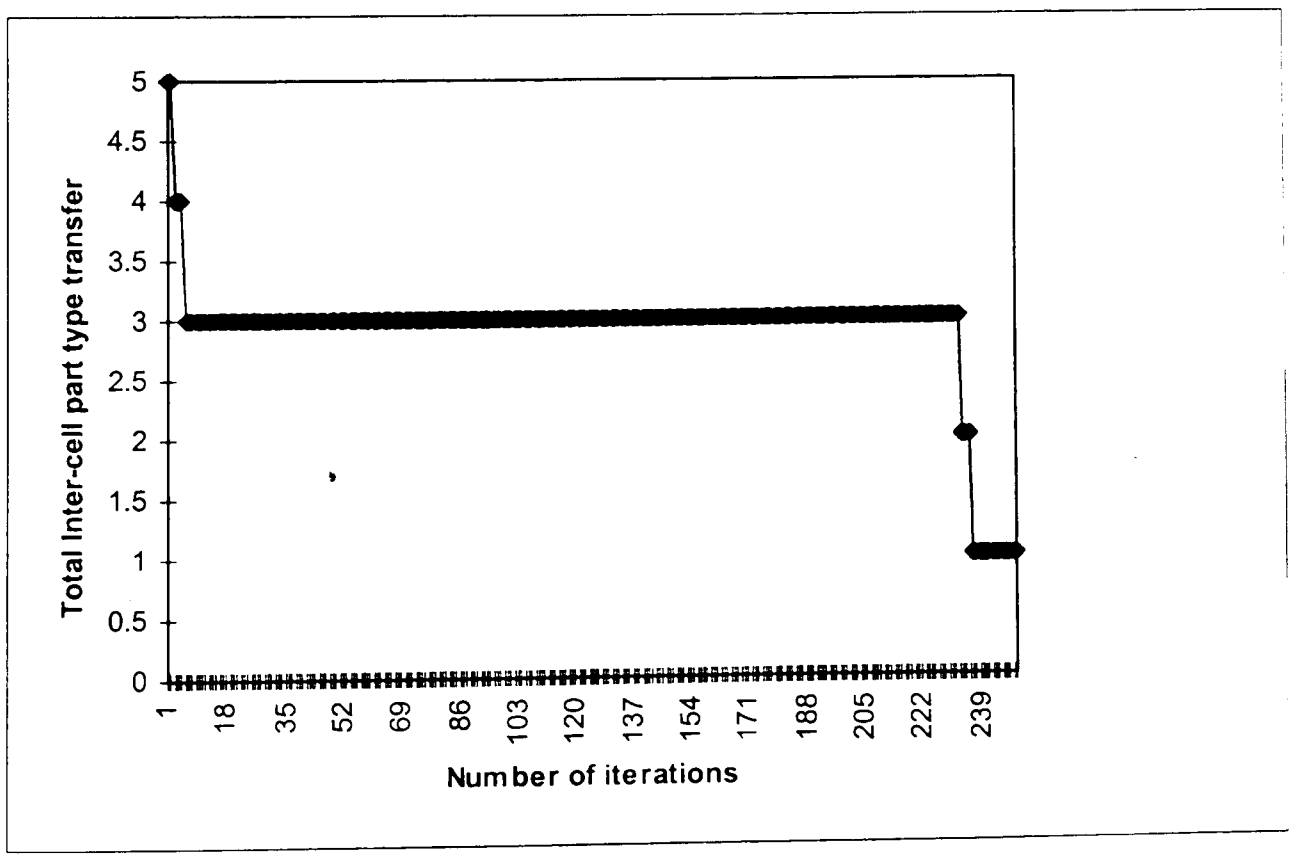


Figure 7.10-a. Total inter-cell movement

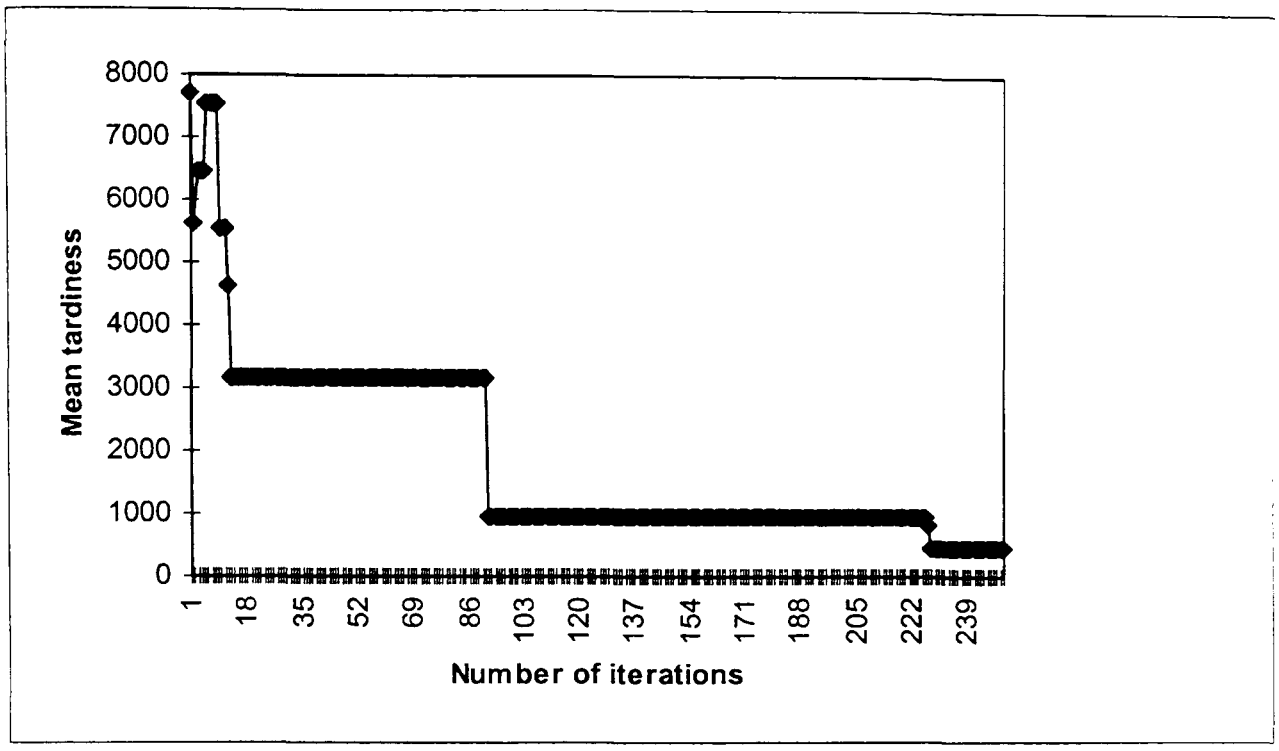


Figure 7.10-b. Mean tardiness performance

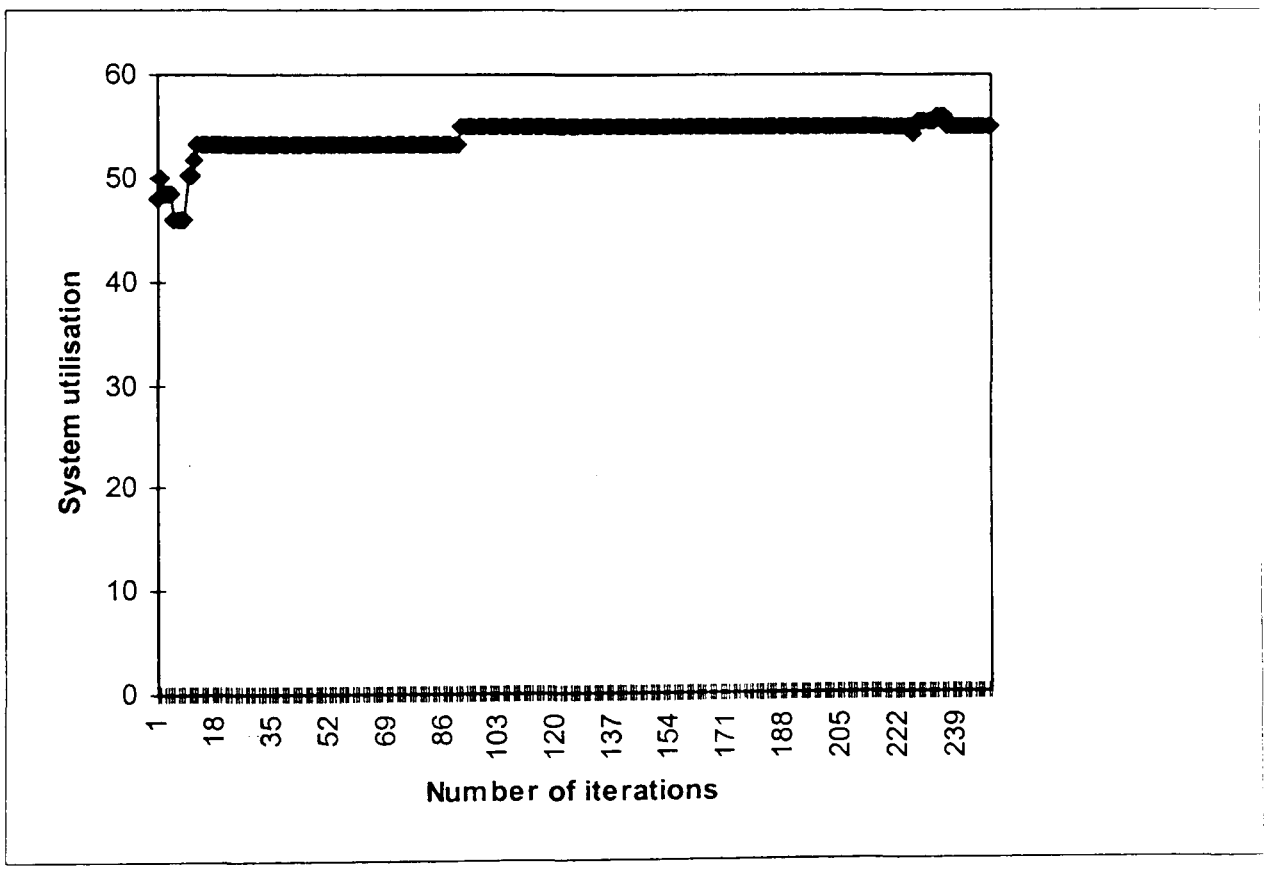


Figure 7.10-c. System utilisation

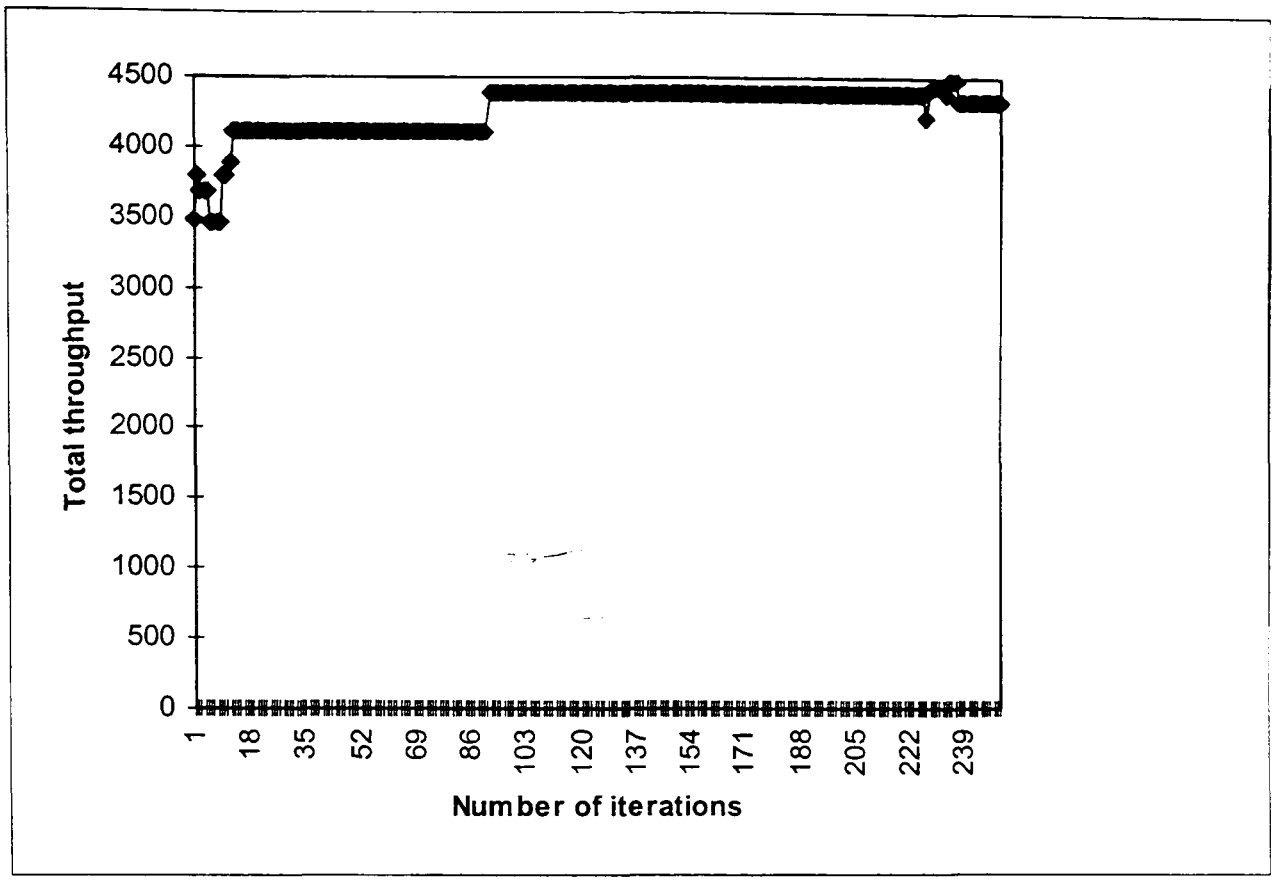


Figure 7.10-d. Total throughput

Figure 7.10 The conversion behaviour of the performance measures considered in the case study

As shown in Figure 7.12 the final routes of the parts are given based on the machines to be used for their processing. These are also the final process plans of the parts that have been determined by the SSM module. The SSM module started with the generic part process plans and selected the suitable machines based on the shop conditions (simulation process) successfully that defines the final process plans and produced schedules for the given manufacturing environment. Consequently, the integration of process planning and loading is achieved.

opt_load.txt - Notepad

File Edit Search Help

```
==================================================
PROGRAM RE-MOCL

Multiple Objective Tabu Search Based Integrated
System for Loading CMS

==================================================

Optimum Part to Cell Assignments
--------------------------------
 Part Type->1 { 1 (1), 2 (1), 4 (1)}
 Part Type->2 { 1 (4), 2 (4), 3 (4)}
 Part Type->3 { 5 (3), 6 (3), 7 (3)}
 Part Type->4 { 8 (2), 5 (2)}
 Part Type->5 { 7 (2), 4 (2), 5 (2)}
 Part Type->6 { 8 (4), 6 (4), 7 (4)}
 Part Type->7 { 8 (1), 9 (1), 10 (1)}
 Part Type->8 { 9 (2), 10 (2), 11 (2)}
 Part Type->9 { 5 (2), 1 (2), 2 (2)}
 Part Type->10 { 3 (2), 4 (2)}
 Part Type->11 { 5 (4), 6 (4), 9 (4)}
 Part Type->12 { 10 (1), 8 (1), 9 (1)}
 Part Type->13 { 5 (4), 8 (4), 10 (4)}
 Part Type->14 { 8 (4), 7 (4), 5 (4)}
 Part Type->15 { 1 (1), 2 (1)}
 Part Type->16 { 3 (2), 4 (2)}
 Part Type->17 { 6 (3), 7 (3), 8 (4)}
 Part Type->18 { 8 (2), 9 (2), 10 (2), 11 (2)}
 Part Type->19 { 5 (3), 2 (3)}
 Part Type->20 { 7 (1), 8 (1), 9 (1)}

Some Performance Indicators

Total Intercell Traffic          =1.0
Mean Tardiness in the System     =461.42
Overall System Utilisation       =55.04
Total Troughput from the System =4334.0
```

Figure 7.11 Part assignment scenario generated by the loading system

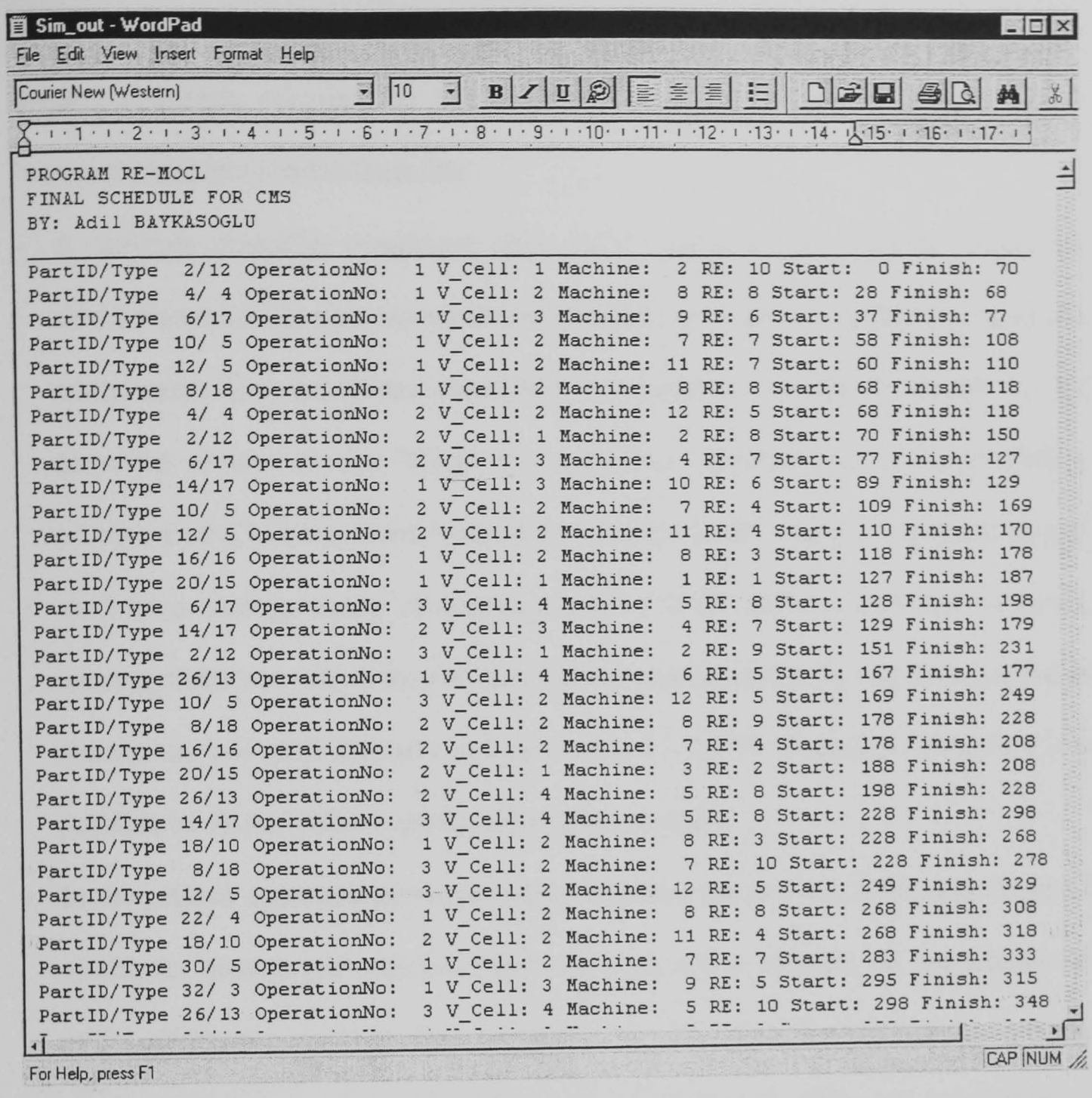


Figure 7.12 A part of the production schedule generated by the loading system

7.5 CONCLUSIONS

An integrated system for loading CMSs is proposed and explained in detail with an application example in this chapter. The proposed system constitutes the loading module of the multiple objective decision-support framework that can evaluate performance of the CMS and gives an indication for reconfiguration action. It is able to integrate process planning and loading decisions via a parametric simulation optimisation strategy that is formally represented in a goal programming structure.

The problem modelling and solution strategy implemented in the present CMS loading approach, contains the following distinguishable features which have not been not considered in the literature:

- A multiple objective simulation optimisation approach is proposed where the simulation model is a parameter that is modified according to different part-to-cell assignments (i.e. parametric simulation optimisation strategy). Based on the literature review of this thesis no equivalent approach for solving loading problems of CMS has been found (see Chapter-2). By using the present hybrid modelling strategy many characteristics of a CMS such as part arrival times, queuing times etc. which are not easy to formulate analytically can be modelled in simulation, and optimisation can be achieved by the TS algorithm. Therefore it is possible to obtain realistic results for implementation.
- In the present approach part-processing requirements are not defined in terms of machines; generic part process plans are used which provide an opportunity to utilise alternative machines for part processing. By using this strategy, a seamless integration of process planning with loading is achieved. Although integration of process planning and scheduling was considered in the literature (Chryssolouris and Chang, 1985, Carvalho and Gindy, 1995), it has not been considered in solving CMS loading problems. Achievement of this integration is a result of using REs in problem modelling. REs can define part processing requirements and machine capabilities in the same language. Therefore one can create a simulation model that dynamically matches part-processing requirements with suitable machines by considering the current status of the CMS (in a simulation study). The RE based scheduling strategy used in this research was proposed by Gindy and Saad (1996). It was shown in their research that RE-based scheduling is more

advantageous than machine-based scheduling in achieving higher manufacturing system performance. However, RE-based scheduling may demand more material handling. It can be concluded that, due to using the same language (RE) in defining part processing requirements and machine capabilities, integration of loading (*part to cell assignment + cell scheduling*) and process planning can be achieved within a simulation optimisation framework such as the one presented in this chapter.

- In the present model, the possibility of inter-cell movement is considered. This issue has not been addressed in the past research (Greene and Sadowski, 1983,1986, Elmaraghy and Gu, 1989). Research on CMS has generally concentrated on the cell formation problem, the CMS loading problem has not receive enough attention. Therefore the current research may be useful in filling this gap.

The paper published by Greene and Sadowski (1986) is the only available paper which presents a mathematical modelling approach for CMS loading problems (i.e. simultaneously determining part to cell assignment and cell scheduling) (see also Chapter 2-Section 2.6). Therefore it may be useful to compare characteristics of their model with the present model. In their model the following assumptions were made: A part can only be assigned to one cell and existence of at least one such cell for all parts is assumed, no cell has more than one machine of any type, cells do not have to have a machine of every type, part routes are machine based and fixed, a part type can visit any machine a maximum of once, additionally their dynamic characteristics such as part arrivals, queue lengths etc., were not considered. As explained above, none of these assumptions are made in the present model. Therefore CMS environments can be presented more realistically. Consequently better loading

solutions can be obtained. They also concluded in their paper that their model is suitable only for very small problems due to the number of variables necessary for modelling. The present model does not require many variables because it uses a hybrid-modelling approach and only part-to-cell assignments constitute model variables. Scheduling is achieved by simulation. However, the present model demands high computational power. It took 123 minutes to solve the test problem that has 20 part types, 12 machines, 4 cells and 11 REs.

Several other test problems have also been solved by the proposed loading system. One more example with relatively bigger size is shown in Appendix VII. However, more computational work can be useful to estimate the quality of the generated solutions in relation to maximum number of iterations. It may also be useful to investigate the effect of changing the orders of objectives on the quality of generated solutions. These issues require comprehensive computational experiments with the proposed model therefore can be considered as future work.

The results of this study shows that it can be possible to improve and/or sustain the performance of CMSs facing changing production requirements through loading mechanisms like the one presented here. Implementation of such mechanisms is also useful while making decisions about CMS reconfiguration, because they can assess the performance of CMS facing changing production requirements. A computer program named as RE_MOCL (Resource Elements based Multiple Objective Cell Loading) has been developed with C/C++ and SIMAN to implement the loading module.

CHAPTER EIGHT

8. DEVELOPMENT OF MULTIPLE OBJECTIVE RECONFIGURATION MODULE

8.1 INTRODUCTION

In this chapter the reconfiguration module of the integrated decision support framework is presented. Reconfiguration decisions are known to be complex decision making problems (Rheault *et. al.,* 1995). It is necessary to consider the reconfiguration of cellular manufacturing systems (CMS) in some production periods due to unsatisfactory performance levels (high tardiness, low throughput etc.) which is related to changing production requirements. As discussed in Chapter 2 performance problems are common in manufacturing environments due to constantly changing needs. The sensitivity of CMSs to changes is also very well known (Sasani, 1990, Kochikar and Narendran, 1998, Seifoddini and Djassemi, 1996, 1997). Therefore, it is necessary to find some ways to reconfigure CMSs to reduce or even to eliminate performance deterioration under changing manufacturing environments. However, there is not a known strategy to reconfigure CMSs as discussed in Chapter 2. One logical approach can be the use of the virtual manufacturing cell (VC) concept. This concept was proposed by The National Bureau of Standards (NBS) (McLean *et. al.*, 1982). It is based on the belief that the factory of the future could be served by a hierarchical structure that could dynamically alter its subsystem allocations as requirements changed. This concept is similar group technology where

job families are processed in manufacturing cells. The major differences between a virtual cell (VC) and group technology is in the dynamic nature of the virtual manufacturing cell; whereas the physical location and identity of traditional group technology cells is fixed, the VC is not fixed and will vary with changing requirements. The VC is a powerful concept, which allows the flexible reconfiguration of shop floors in response to changing requirements.

Drolet and her colleagues, (Drolet *et. al.,* 1989,1990, Montreuil *et. al.,* 1992) further studied the VC concept and developed algorithms for creating and scheduling them. In their Virtual Cellular Manufacturing Systems (VCMS) framework, when a job order needs a set of workstations to be put together, a VC controller takes over the control of these workstations and makes communication possible between them. A workstation is either a member of a pool of available workstations or member of a VC. VCs are created by using minimum-spanning-tree algorithms (MST). The candidate machine tool set for the most prioritised job order is determined first, then the sub-set which has the minimum travelling distance score is determined by the MST algorithm, finally a VC controller is created and assigned to the job order. The same procedure is iterated for the remaining job orders. If a machine is member of more than one VC, the most prioritised job has the priority for processing. However, there are some drawbacks in their way of creating VCs that can negatively affect the performance of CMS. The possibility of using other objectives for creating VCs was not considered. If multiple jobs are required to be processed simultaneously, the minimisation of travel distance for each job separately might not result in a good or competitive shop performance. A comparison with FIFO scheduling was made by Chatterjee (1992) and no performance superiority was observed. The processing

requirements of parts are defined based on machine requirements and the possibility of using alternative machines was not considered in their framework. The realisation of machine capabilities can be beneficial for creating VCs. As a final remark there is no mechanism to avoid machine sharing when possible, in fact, machine sharing[1] increases the complexity of the control considerably.

There is no explicit decision making for reconfiguration in VCMS. Reconfiguration is a natural process and happens continuously when a job order arrives. As concluded by Drolet *et. al.*(1995), their framework can be applied to highly automated factories of the future which have not more than 20 highly productive machines and a flexible automated material handling system. However, by using virtual cell definition it is also possible to make reconfiguration decisions explicitly and to find globally optimal solutions. In the present approach, reconfiguration decisions are made periodically in discrete production periods. When the CMS is found not to perform well after loading then a reconfiguration action is triggered. Therefore reconfiguration is an explicit decision and it is based on performance indicators determined by the loading system. Reconfiguration is achieved by generating virtual cells. The virtual cell formation is logically similar to McLean *et. al.* (1982)'s and Drolet *et. al.* (1989) virtual cell definition i.e. while forming VCs only cell memberships of machines are updated, machines are not dislocated physically. However in the present approach, formation of virtual cells is decided based on the system performance measures in discrete time periods and while forming them, alternative machines for part processing are considered and machine sharing is avoided when possible.

[1] Assigning a machine to more then one virtual cell.

The following definition is used for virtual cells in this thesis (Gindy, 1997, Baykasoglu *et. al.,* 1998-b): *A virtual cell is an objective driven logical grouping of manufacturing resources*. In this chapter, a hybrid mathematical programming-simulation optimisation model is proposed to generate virtual cell configurations. It is also aimed to understand whether it is possible to improve the performance of CMS by using virtual manufacturing cells.

8.2 THE RECONFIGURATION STRATEGY

In this work reconfiguration is performed by creating virtual cells based on the following strategy, VC creation is decided by the higher level planning systems i.e. production planning system. If the current set of virtual cells cannot cope with the new production requirements launched by the production planning department (as part lists) then a new set of VCs is generated. Multiple performance measures and capabilities of available machine tools as REs are considered in VC generation. Interaction between VCs (inter-cell movement or machine sharing) is also minimised. Therefore the framework can easily be applied to traditional CMSs.

As summarised in Chapter 3 the following steps are followed for reconfiguration. The performance of the current configuration is evaluated by the simulation-optimisation based loading system for the coming production period and a decision with regard to reconfiguration is made. If the performance of the current configuration was found unsatisfactory then reconfiguration is done before actual production starts. Based on the capability of the available resources and production requirements of parts in the coming production period, candidate virtual cell scenarios which satisfy the previously defined constraints (e.g. cell size etc.) are

generated and the one which best satisfies the performance measures (inter-cell movement, mean tardiness, overall utilisation and total throughput) is adopted. The model is formally represented in a pre-emptive goal programming framework and solved by the multiple objective Tabu search algorithm proposed in Chapter 5.

8.3 MATHEMATICAL MODELLING OF VIRTUAL CELL FORMATION PROBLEM

The virtual cell formation problem is formally represented as a pre-emptive goal programming model with the following performance measures; cell interaction, mean tardiness in the system, mean system utilisation and average throughput from the system. The first objective function value is obtained from an analytical equation that was presented in Chapter 7, and the last three are obtained from the parametric simulation/scheduling module that was also explained in Chapter 7.

The pre-emptive goal programming model for virtual cell formation is given as follows:

$$\text{lexmin}\{(d_c^+),(d_t^- + d_t^+),(d_u^- + d_u^+),(d_q^- + d_q^+)\} \tag{8.1}$$

Constraints

$$\sum_{i=1}^{m} \sum_{o=1}^{NOP_i-1} \sum_{k=1}^{g_{\max}} \left| X_{iok} - X_{i(o+1)k} \right| + d_c^- - d_c^+ = goal\_1 \tag{8.2}$$

$$\boldsymbol{MT} + d_t^- - d_t^+ = goal\_2 \tag{8.3}$$

$$\boldsymbol{MU} + d_u^- - d_u^+ = goal\_3 \tag{8.4}$$

$$\boldsymbol{TT} + d_q^- - d_q^+ = goal\_4 \tag{8.5}$$

$$\sum_{k=1}^{g_{\max}} X_{iok} = 1 \quad \forall i,o \tag{8.6}$$

$$\sum_{k=1}^{g_{\max}} Y_{jk} = 1 \quad \forall j \tag{8.7}$$

$$M_{\max} Z_k - \sum_{j=1}^{n} Y_{jk} \geq 0 \quad \forall k \tag{8.8}$$

$$\sum_{j=1}^{n} Y_{jk} - M_{\min} Z_k \geq 0 \quad \forall k \tag{8.9}$$

$$\sum_{j=1}^{n} Y_{jk} p_{j(R_{io})} - X_{iok} \geq 0 \quad \forall i,o,k \tag{8.10}$$

$$Y_{jk}, X_{iok}, Z_k \in \{0,1\} \quad \& \quad d_c^-, d_c^+, d_t^-, d_t^+, d_u^-, d_u^+, d_q^-, d_q^+ \geq 0 \quad \forall i,j,k,o \tag{8.11}$$

$$g_{\min} = \lceil n / M_{\max} \rceil \leq g \leq \lfloor n / M_{\min} \rfloor = g_{\max} \tag{8.12}$$

where:

j: Machine index

$p_{j(Rio)}$: If RE corresponding to the o'th operation of part i is available on machine j 1, otherwise 0.

g: Number virtual cells

g_{max}: Maximum number of virtual cells

g_{min}: Minimum number of virtual cells

M_{min}: Minimum number of machines in a virtual cell

M_{max}: Maximum number of machines in a virtual cell

X_{iok}: 1 if o'th operation (corresponding RE) of part i is assigned to virtual cell k, 0 otherwise.

Y_{jk}: 1 if machine j is assigned to virtual cell k, 0 otherwise

Z_k: 1 if virtual cell k is formed, 0 otherwise

For the rest of the notation refer to Chapter 7-Section 7.3.1

Equation 8.1 represents the lexicographical order of the deviational variables to be minimised that is similar to objective function that has been presented in Chapter 7. Equation 8.2 represents the first goal constraint (cell interaction). For a concurrent virtual cell creation scenario this equation calculates the total number of inter-cell movements. Equations 8.3, 8.4 and 8.5 are second, third and fourth goal constraints respectively. The numeric values of ***MT*** (mean tardiness in CMS), ***MU*** (mean system utilisation of CMS) and ***TT*** (total throughput from the CMS) are obtained from the simulation/scheduling module. Equation 8.6 ensures that each operation of a part type is assigned to only one virtual cell. Equation 8.7 ensures that every machine is assigned to only one virtual cell. Equations 8.8 and 8.9 limit the number of machines assigned to each virtual cell if it is formed. Equation 8.10 ensures that, if an operation of a part is assigned to a virtual cell the RE corresponding to that operation is available in that cell. Maximum and minimum number of virtual cells are determined by using Equation 8.12. The solution of the above model is achieved by the multiple-objective Tabu search algorithm that has been developed in Chapter 5. The problem specific adjustments to the original Tabu search algorithm in order to improve its effectiveness are explained in the following section. The above model may produce some cells with machine assignment but no part assignment (due to excess capacity, non-required REs in some periods, etc.). These cells are merged and considered as a pool cell.

As it can be seen from the above formulation, the objectives of the virtual cell formation (i.e. reconfiguration) are similar to the loading model's (see Chapter 7) objectives and constraints are similar to the initial cell formation model's (see Chapter 6) constraints. This fact explains the logic behind the virtual cell formation process. The main purpose of the virtual cell formation process is to improve the operational performance (tardiness, throughput etc.) of the CMS in the existing production period by satisfying design constraints (i.e. cell size etc.). In the virtual cell formation step, demand in the current loading period is taken into account, therefore the time span is limited to that period (it may be longer if demand does not change or affect performance in the following loading periods). However, in the initial cell formation process, satisfaction of the design objectives is the main purpose. In the initial cell formation step gross demand for the entire planning horizon is taken into account to determine the overall structure of the CMS system.

8.3.1 Application of Multiple Objective TS algorithm to Solve Virtual Cell Formation Model

The multiple-objective Tabu Search (TS) algorithm that has been developed in Chapter 5 is employed to solve the virtual cell formation model. However, a number of problem-specific features are incorporated into the original TS algorithm to improve its convergence speed. The original TS algorithm can also be used without any change. The following problem-related modifications are made:

Initial Solution:

The worst performing virtual cell configuration in the preceding production period constitutes the initial solution.

Generation of neighbourhood solutions:

Instead of using the neighbourhood generation functions that were presented in Chapter 4-Section 4.2.1., the following movement strategy is applied for generating neighbourhood solutions from the current solution (the current solution is the initial solution in the first iteration of Tabu search, in the subsequent steps it is the best neighbourhood solution):

- *Step-1*: (Determination of cell sets) Determine the set of cells which has more machines than M_{min} and less than M_{max} and call this set as C_{mm}. Determine the set of cells which has M_{min} number of machines and call this set as C_{qm}. Determine the set of cells which has M_{max} number of machines and call this set as C_{xm}. If the total number of cells is less than g_{max} in the current configuration then, call the remaining empty set as C_{em}.
- *Step-2*: Select a non-empty cell randomly (call it as feeder cell). Then, select M_{min} number of machines randomly from this cell.
- *Step-3*: Select another cell randomly which does not belong to C_{xm} set (target cell). If the selected cell belongs C_{em} set then assign all M_{min} number of machines to that cell. If the selected cell belongs C_{qm} or C_{mm} set then, assign only one of the randomly determined machine to that cell. Then update the cell set statues and pick up another cell randomly from C_{qm} or C_{mm} sets and assign one machine then update cell set statutes and continue the same procedure until all machines are assigned.
- *Step-4*: Check all cells for the parts that have operations assigned to them but they cannot satisfy these operations (i.e. they do not have corresponding RE because machine reassignments). Collect these parts in a set and call this set as P_u. If P_u is

empty then randomly select a part from a randomly determined non-empty cell and put it into P_u.

- *Step-5*: Starting from the first operation in the processing sequence of the first part in the set P_u, assign it to the randomly selected non-empty cell if it can supply the required RE for this operation, if not select another non-empty cell randomly which can supply it. Continue with the next operation in the processing sequence, if the RE required by this operation can be supplied by the current cell assign it to this cell otherwise choose another cell randomly which can supply it. Continue to this process until all operations of the part are assigned. Apply the same procedure to the other parts in the set P_u.
- *Step-6*: Iterate Steps 1 to 5 until the required number of neighbourhood solutions are generated from the current solution.

By applying the above heuristic movement strategy it is possible to generate feasible solutions quickly and it is an effective strategy for achieving the first goal.

Tabu list:

Indices of randomly selected and reassigned machines and part pairs are put into the *tabu list.*

Aspiration criteria:

Any move that improves the best known solution is accepted, even if the move is tabu (see also Chapter 6-Section 6.3 for more details about aspiration criteria).

8.4 EXPERIMENTAL WORK†

The best loading scenario that was obtained by the loading module in Chapter 7 was considered not good enough to satisfy performance targets set by the decision-maker (the decision-maker goals were set to 0 for inter-cell part type transfer, 0 for mean tardiness, 65% for system utilisation, and 5000 for total throughput). The performance measures obtained from the loading module with the given virtual cell configuration were 1 inter-cell part type transfer, 461.42 mean tardiness, 55.04% system utilisation, and 4334 total throughput. In particular, mean tardiness, system utilisation and total throughput goals were candidates for further improvement. The decision to improve the performance measures by reconfiguration (i.e. regenerating VCs to better satisfy the current production requirements) is taken. All the technical data necessary for reconfiguration were given in section 7.4 of Chapter 7.

The number of iterations in the reconfiguration module was 250 and in each iteration three neighbourhood solutions were generated. This required 750 simulation experiments. Every time a feasible candidate neighbour is generated by the Tabu search algorithm the simulation experiment file is automatically regenerated for this solution to reflect the new scenario. The time required to finish 250 iterations was 125 minutes on a Pentium 200 PC with 32MB RAM. The computational time is directly dependent on the size of the problem, number of iterations, the number of neighbourhood solutions generated in each iteration and the computational power available.

† The experimental work of this section is a continuation of the experimental work that was presented in Chapter 7-Section 7.4.

The results obtained from the experimental study are shown in Figures 8.1-8.3. The interpretation of the results show that the inter-cell movement goal and the mean tardiness goal are totally satisfied after generating a new set of virtual cells. The system utilisation is improved from 55.04% to 60.318% in the new virtual cell configuration which is now nearer to the decision-maker's goal which was 65%. The total throughput goal is improved from 4334 to 4970 in the new virtual cell configuration which is nearer to decision-maker's goal of 5000. The results obtained from this experiment show that performance measures can be improved by using virtual cells as a reconfiguration strategy.

In Figure 8.1 the conversion behaviour of the performance measures considered in this research work are shown. Figure 8.1-a displays the best neighbourhood solutions for the inter-cell part type transfer objective and how the developed system reduced inter-cell movement to zero in iteration number 2 (see also zoomed section between iteration numbers 1 to 3 in Figure 8.1-a and iteration numbers between 14 to 55 in Figure 8.1-b). The purpose of these zoomed parts is to help in understanding why the other performance measures, 'utilisation' and 'throughput' have changed to worse values between these iteration numbers. Because, 'inter-cell movement' and 'mean tardiness' improved during these iterations. In short, objectives with lower priorities may deteriorate temporarily while objectives with higher priorities are improving. While more important objectives are not improving, less important objectives continue to improve. All objectives can also improve simultaneously. As a result all objectives are optimised simultaneously.

The outputs of the reconfiguration module are the new set of virtual cells, part to cell assignment and schedules for virtual cells that are presented in Figures 8.2 and 8.3.

The performance improvement by the new virtual cell configuration is also shown in Figure 8.4. This figure depicts the best possible performance without generating a new set of virtual cells and the best possible performance after generating a new set of virtual cells. As can be seen from the figure all performance measures are considerably improved.

Further experimental work is carried out with another, demand scenario where the above final configuration constitutes the initial configuration. Detailed explanations and results are given in Appendix VII. The results obtained in Appendix VII has also proved that VCs are a valid candidate for the CMS reconfiguration. However, further research and extensive computational study can be useful to determine the optimality of the generated solutions and to determine the best possible set of TS parameters (number of iterations etc.) in order to minimise the computational time requirements in obtaining an acceptable solution.

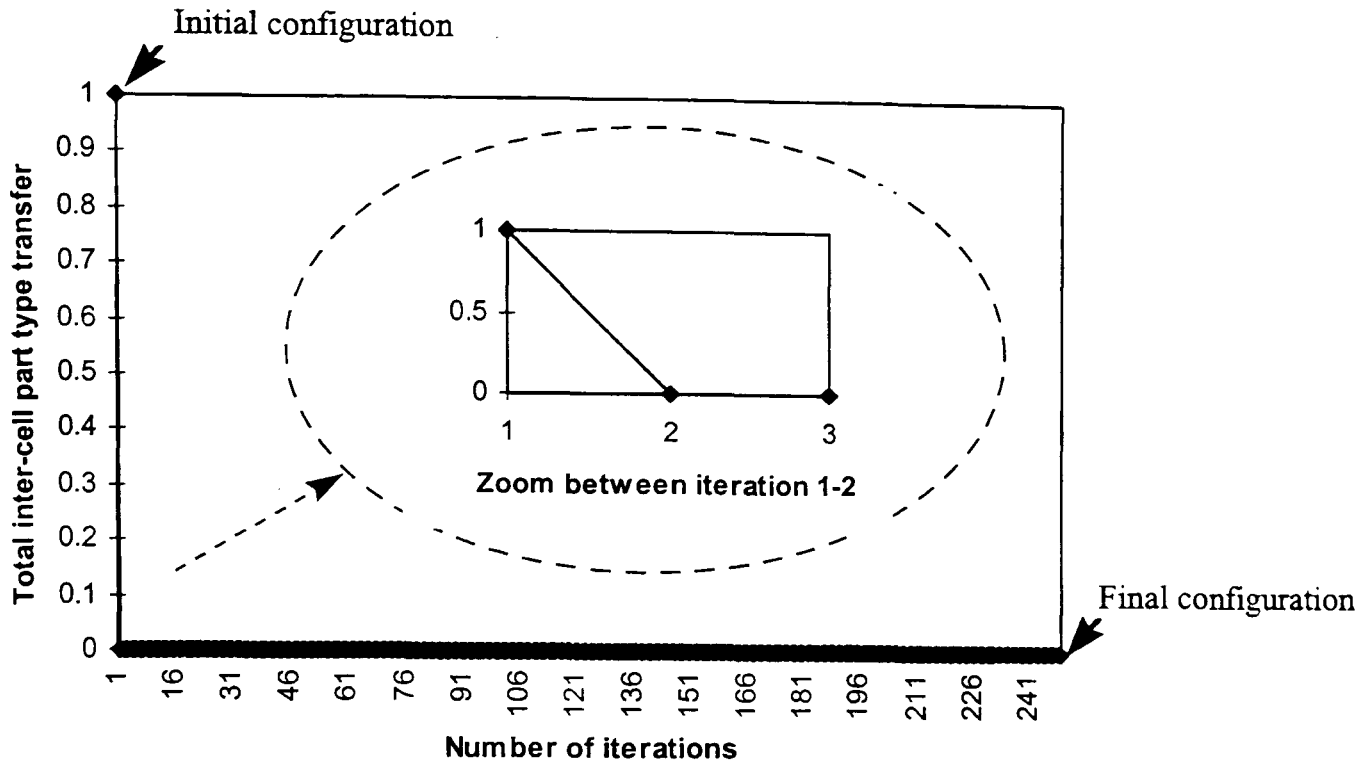


Figure 8.1-a. *Total inter-cell movement*

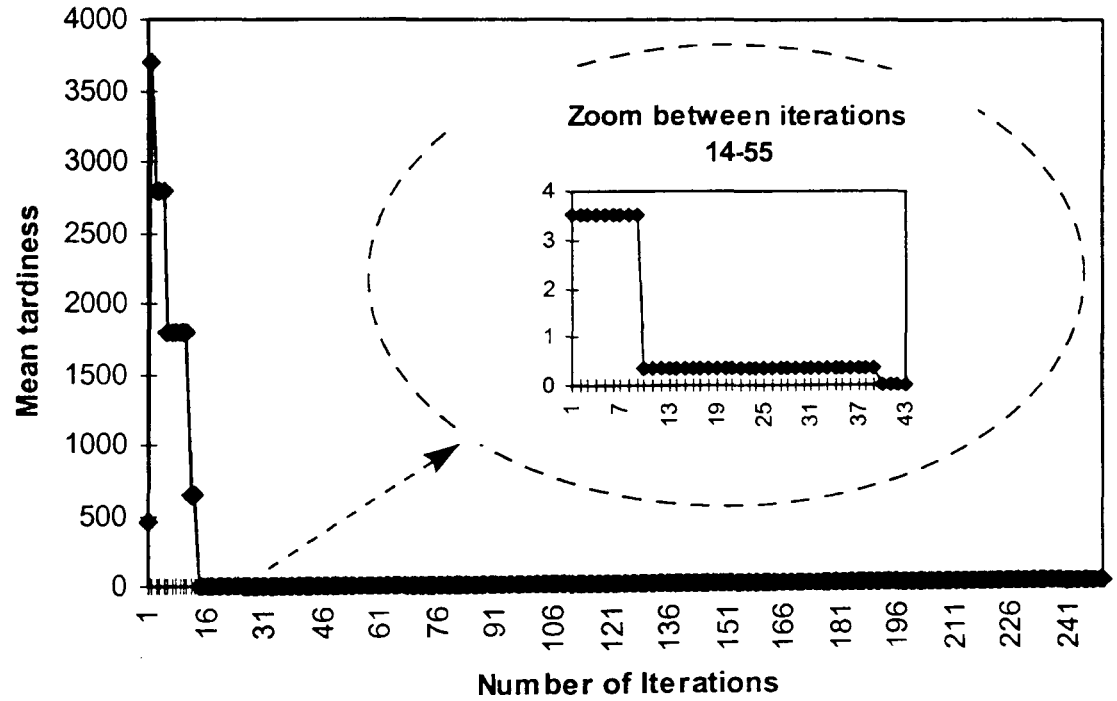


Figure 8.1-b. *Mean tardiness*

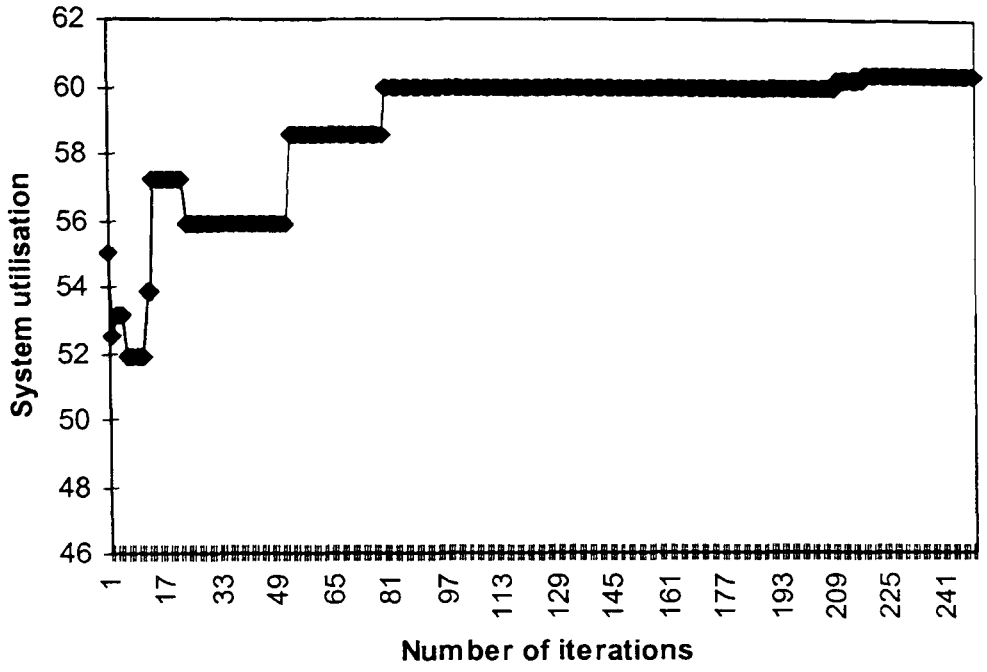


Figure 8.1-c. *System utilisation*

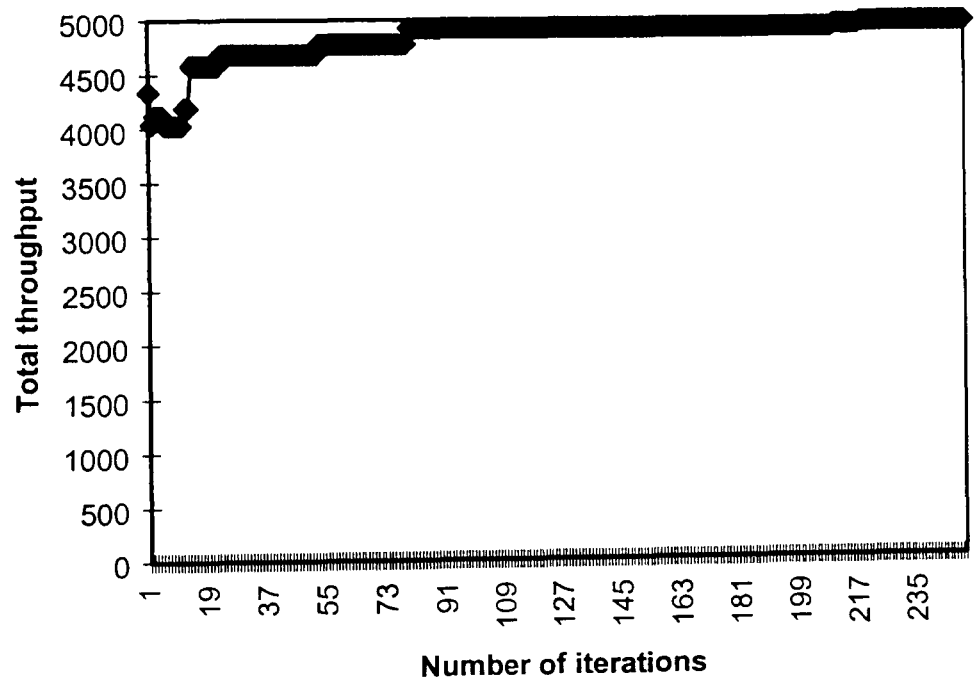


Figure 8.1-d. *Total throughput*

Figure 8.1 The conversion behaviour of the performance measures

OUTPUT OF RECONFIGURATION PROGRAM (NEW VCs & PART ASSIGNMENT)

VCs after Reconfiguration

Virtual Cell->1
Machine->5
Machine->6
Machine->8
Machine->11
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->3, RE->4, RE->5, RE->6, RE->7, RE->8, RE->9, RE->10, RE->11
Virtual Cell->2
Machine->7
Machine->12
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->4, RE->5, RE->7, RE->10
Virtual Cell->3
Machine->1
Machine->3
Machine->4
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->4, RE->5, RE->6, RE->7
Virtual Cell->4
Machine->2
Machine->9
Machine->10
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->3, RE->4, RE->6, RE->7, RE->8, RE->9, RE->10, RE->11

Part assignments for new VCs

Part Type->1 { 1 (1), 2 (1), 4 (1)}
Part Type->2 { 1 (4), 2 (4), 3 (4)}
Part Type->3 { 5 (3), 6 (3), 7 (3)}
Part Type->4 { 8 (1), 5 (1)}
Part Type->5 { 7 (3), 4 (3), 5 (3)}
Part Type->6 { 8 (4), 6 (4), 7 (4)}
Part Type->7 { 8 (4), 9 (4), 10 (4)}
Part Type->8 { 9 (1), 10 (1), 11 (1)}
Part Type->9 { 5 (3), 1 (3), 2 (3)}
Part Type->10 { 3 (4), 4 (4)}
Part Type->11 { 5 (1), 6 (1), 9 (1)}
Part Type->12 { 10 (4), 8 (4), 9 (4)}
Part Type->13 { 5 (1), 8 (1), 10 (1)}
Part Type->14 { 8 (1), 7 (1), 5 (1)}
Part Type->15 { 1 (3), 2 (3)}
Part Type->16 { 3 (1), 4 (1)}
Part Type->17 { 6 (1), 7 (1), 8 (1)}
Part Type->18 { 8 (1), 9 (1), 10 (1), 11 (1)}
Part Type->19 { 5 (1), 2 (1)}
Part Type->20 { 7 (1), 8 (1), 9 (1)}

Figure 8.2 Output of reconfiguration module as the new set of virtual cells and corresponding part assignment

Sim_out - WordPad

File Edit View Insert Format Help

Courier New (Western) 10

```
PartID/Type  2/12 OperationNo:  1 V_Cell: 4 Machine:  2 RE: 10 Start:  0 Finish: 70
PartID/Type  4/ 4 OperationNo:  1 V_Cell: 1 Machine:  5 RE: 8 Start: 28 Finish: 68
PartID/Type  6/17 OperationNo:  1 V_Cell: 1 Machine:  6 RE: 6 Start: 37 Finish: 77
PartID/Type  8/18 OperationNo:  1 V_Cell: 1 Machine:  8 RE: 8 Start: 48 Finish: 98
PartID/Type 10/ 5 OperationNo:  1 V_Cell: 3 Machine:  3 RE: 7 Start: 58 Finish: 108
PartID/Type  2/12 OperationNo:  2 V_Cell: 4 Machine:  2 RE: 8 Start: 70 Finish: 150
PartID/Type  4/ 4 OperationNo:  2 V_Cell: 1 Machine:  6 RE: 5 Start: 77 Finish: 127
PartID/Type  6/17 OperationNo:  2 V_Cell: 1 Machine:  5 RE: 7 Start: 77 Finish: 127
PartID/Type  8/18 OperationNo:  2 V_Cell: 1 Machine:  8 RE: 9 Start: 98 Finish: 148
PartID/Type 12/ 5 OperationNo:  1 V_Cell: 3 Machine:  3 RE: 7 Start: 108 Finish: 158
PartID/Type 14/17 OperationNo:  1 V_Cell: 1 Machine:  6 RE: 6 Start: 127 Finish: 167
PartID/Type 16/16 OperationNo:  1 V_Cell: 1 Machine:  5 RE: 3 Start: 127 Finish: 187
PartID/Type 20/15 OperationNo:  1 V_Cell: 3 Machine:  1 RE: 1 Start: 127 Finish: 187
PartID/Type  6/17 OperationNo:  3 V_Cell: 1 Machine:  8 RE: 8 Start: 148 Finish: 218
PartID/Type 18/10 OperationNo:  1 V_Cell: 4 Machine:  2 RE: 3 Start: 150 Finish: 190
PartID/Type 10/ 5 OperationNo:  2 V_Cell: 3 Machine:  3 RE: 4 Start: 158 Finish: 218
PartID/Type 26/13 OperationNo:  1 V_Cell: 1 Machine:  6 RE: 5 Start: 167 Finish: 177
PartID/Type 14/17 OperationNo:  2 V_Cell: 1 Machine: 11 RE: 7 Start: 167 Finish: 217
PartID/Type 22/ 4 OperationNo:  1 V_Cell: 1 Machine:  5 RE: 8 Start: 187 Finish: 227
PartID/Type  2/12 OperationNo:  3 V_Cell: 4 Machine:  2 RE: 9 Start: 190 Finish: 270
PartID/Type 18/10 OperationNo:  2 V_Cell: 4 Machine:  9 RE: 4 Start: 191 Finish: 241
PartID/Type 16/16 OperationNo:  2 V_Cell: 1 Machine: 11 RE: 4 Start: 217 Finish: 247
PartID/Type 24/16 OperationNo:  1 V_Cell: 1 Machine:  8 RE: 3 Start: 218 Finish: 278
PartID/Type 12/ 5 OperationNo:  2 V_Cell: 3 Machine:  3 RE: 4 Start: 218 Finish: 278
PartID/Type 10/ 5 OperationNo:  3 V_Cell: 3 Machine:  9 RE: 5 Start: 219 Finish: 299
PartID/Type  8/18 OperationNo:  3 V_Cell: 1 Machine:  5 RE: 10 Start: 227 Finish: 277
PartID/Type 22/ 4 OperationNo:  2 V_Cell: 1 Machine:  6 RE: 5 Start: 228 Finish: 278
PartID/Type 28/ 6 OperationNo:  1 V_Cell: 4 Machine:  2 RE: 8 Start: 270 Finish: 320
PartID/Type 26/13 OperationNo:  2 V_Cell: 1 Machine:  5 RE: 8 Start: 277 Finish: 307
PartID/Type 14/17 OperationNo:  3 V_Cell: 1 Machine:  8 RE: 8 Start: 278 Finish: 348
PartID/Type 20/15 OperationNo:  2 V_Cell: 3 Machine:  3 RE: 2 Start: 278 Finish: 298
PartID/Type 24/16 OperationNo:  2 V_Cell: 1 Machine: 11 RE: 4 Start: 278 Finish: 308
PartID/Type 30/ 5 OperationNo:  1 V_Cell: 3 Machine:  3 RE: 7 Start: 298 Finish: 348
PartID/Type 12/ 5 OperationNo:  3 V_Cell: 3 Machine:  4 RE: 5 Start: 299 Finish: 379
PartID/Type  8/18 OperationNo:  4 V_Cell: 1 Machine:  5 RE: 11 Start: 307 Finish: 327
```

NUM

For Help, press F1

Figure 8.3 A small part of production schedule generated automatically for the new set of virtual cells

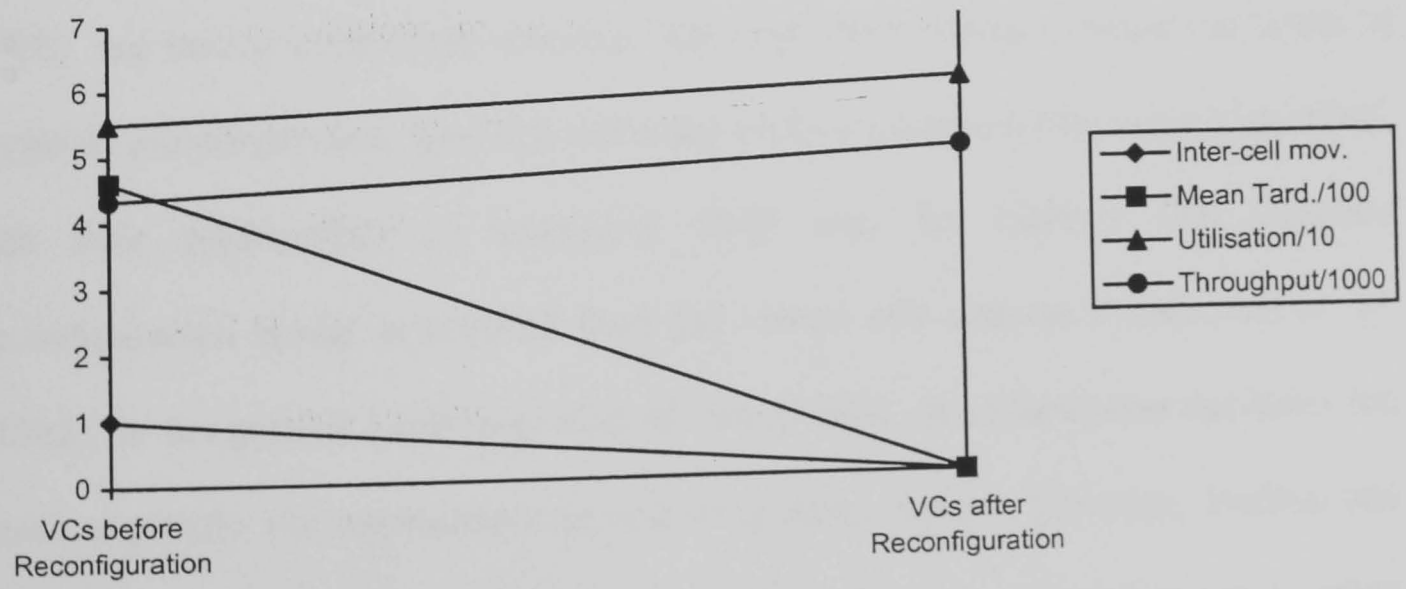


Figure 8.4 Performance improvements after reconfiguration

8.5 CONCLUSIONS

In this chapter, the reconfiguration module of the integrated framework is explained. The virtual cell concept is used as the reconfiguration strategy. A hybrid modelling strategy that is composed of mathematical programming, Tabu search and parametric simulation is implemented for generating virtual cells. The proposed model is tested with the test problem detailed in Chapter 7. The results obtained from this test problem and other tests (one more test study is reported in Appendix VII) showed that the virtual cell concept is a valid option for the reconfiguration of CMS. By using the virtual cell concept, rapid and economic reconfiguration of CMS is possible, because reconfiguration with virtual cells does not require physical dislocation of machines. Implementing virtual cells can prevent performance deterioration of a CMS facing changing production requirements.

As discussed in Chapter 2, reconfiguration of CMS has not received much attention in the literature. Some of the proposed strategies like the 'virtual cellular manufacturing' framework of Drolet *et. al.*, (1995), 'holonic manufacturing' framework (Markus *et. al.*,1996), 'random manufacturing' framework (Iwata *et. al.*, 1996) are mainly controlling schemes. Although their implementation can result in dynamic reconfiguration, they don not make explicit decisions about reconfiguration, and their applicability to traditional CMS may be limited. The proposed reconfiguration model is inspired from the virtual cell concept of McLean *et. al.* (1982). In the present implementation of virtual cells, reconfiguration decisions are made explicitly via interconnecting and integrating, process planning, loading and cell formation activities within a multiple objective simulation-optimisation

framework. The proposed framework can be used for the reconfiguration of traditional as well as modern CMS.

The reconfiguration module is implemented in the C/C++ computer programming language and the SIMAN simulation language on a Pentium 200 PC with 32 MB RAM.

CHAPTER NINE

9. SUMMARY, CONCLUSIONS AND FURTHER RESEARCH

9.1 INTRODUCTION

In this chapter, a brief summary and the main contributions of the work reported in this thesis are presented. Further research areas are also discussed.

9.2 SUMMARY OF THE THESIS

Chapter 1 of the thesis gave an introduction to the research and explained the research objectives and organisation of the thesis.

Chapter 2 presented a detailed review and analysis of the literature related to the topics being studied.

Chapter 3 gave an overview of the proposed multiple objective decision support framework for configuring, loading and reconfiguring cellular manufacturing systems. Brief descriptions about modules (cell formation, loading, reconfiguration, multiple objective optimisation and simulation) of the framework were given and their interconnection has been explained.

Chapter 4 presented the application of Tabu search to the general problem of multiple objective optimisation (Pareto optimality). A computer algorithm and program have

been developed to assess the potential of Tabu search in finding Pareto optimal solutions in multiple objective optimisation. Various test problems from different domains have been solved with the proposed algorithm and results have been compared.

Chapter 5 presented application of Tabu search for solving pre-emptive goal programming models. Pre-emptive goal programming is used in formulating the multiple objective decision making problems studied in this thesis. A Tabu search based algorithm and a computer program have been developed in this chapter to solve pre-emptive goal programming models. Test problems from various domains have been solved with the proposed algorithm in order to evaluate its performance in solving pre-emptive goal programs. The obtained results have been also compared.

In Chapter 6, a multiple-objective simultaneous manufacturing cell formation model was developed. The application of a multiple-objective Tabu search algorithm for the solution of the proposed cell formation model was explained. An example application and comparative work were also carried out to test the efficiency of the proposed model.

In Chapter 7, a simulation-optimisation based integrated model for loading a cellular manufacturing system with multiple objectives was developed. Development of the parametric simulation model was explained. Application of the Tabu search algorithm was shown for solving the proposed loading model. An example application with the prototype model was explained and the model characteristics were compared with some other approaches.

In Chapter 8, the reconfiguration of cellular manufacturing systems by use of virtual cells was explained. The proposed reconfiguration procedure and Tabu search based multiple objective simulation optimisation algorithm were explained. Characteristics of the proposed model were discussed and an example application with the prototype model was described.

9.3 THE RESEARCH CONTRIBUTIONS

Although the conclusions and achievements of this research were discussed specifically at the end of each chapter, in this part of the thesis, an overview of the results of the whole work will be discussed.

a) Multiple Objective Optimisation

Many engineering problems can be represented as optimisation problems. However, it is important to have general-purpose problem and model independent optimisation procedures that can be applied for their solution. During this study, Tabu search (TS) based optimisation algorithms have been developed to solve general purpose Multiple Objective Optimisation (MOO) problems. TS is a widely accepted heuristic optimisation technique. The problem-independent nature of TS makes it a good candidate for engineering optimisation problems. TS has generally been applied to single objective optimisation problems. However, TS has a potential to be directly applied to MOO problems without requiring extra procedures such as weighting and summing up objective functions. This is mainly due to its inherent solution mechanism in which it works with a population of solutions. This observation led to the direct application of TS to the MOO problems.

- First the applicability of TS to the general problem of MOO (i.e. finding Pareto optimal set) was investigated in Chapter 4. Before attempting to use a special technique like TS to find a Pareto optimal solution for a MOO problem, its ability to determine the Pareto optimal solution set needs to be known. Having this in mind a general purpose TS algorithm was developed in Chapter 4, to test the ability of TS in finding Pareto optimal solutions. Results of the comparative study presented in this thesis are as follows:

In the first test study a comparison was made with the GINO software that uses the constraint method (Winston, 1994). The GINO software found 9 Pareto optimal solutions whereas the proposed TS algorithm found 204 Pareto optimal solutions for the given problem (see Chapter 4).

In the second test study a comparison was made with the Murata *et. al.* (1996) genetic algorithm (MOGA). The proposed TS algorithm found 40% more Pareto optimal solutions than MOGA.

In the third test study a comparison was made with the Osyczka and Kundu (1996) genetic algorithm. The maximum number of Pareto optimal solutions found by their algorithm was only 34, whereas the proposed TS algorithm found hundreds of Pareto optimal solutions and determined the exact shape of the trade-off curve for the given problem. In the fourth test study, a comparison was again made with the Osyczka and Kundu (1996) genetic algorithm using a different problem. This time their genetic algorithm found 133 Pareto optimal solutions whereas the proposed TS algorithm found 5964 Pareto optimal solutions. Additionally the

extreme points obtained by the proposed TS algorithm were around 50% better then their extreme points. By using the same test problem, a comparison was also made with the Plain Stochastic Method (PSM). PSM found only 19 Pareto optimal solutions, less than both Osyczka and Kundu (1996) genetic algorithm and the proposed TS algorithm. The results show that the proposed TS algorithm has the ability to solve MOO problems. A C++ computer program was developed during the study to implement the proposed algorithm.

- After showing that TS has the ability to solve MOO problems, a general purpose TS algorithm was developed to solve pre-emptive goal programming models (PGP). PGP is a widely accepted framework for modelling multiple objective decision making problems (Ignizo, 1982). Multiple objective decision making problems of this thesis have been also formulated by using PGP. The proposed TS algorithm was tested with various test problems from literature. In five of the test problems collected from literature (Winston, 1994, Sundaram, 1978, Schniederjans and Kwak, 1982, Ignizo, 1982 and the LINDO manuals (LINDO, 1996)), the optimal solutions were known. The proposed TS algorithm found these optimal solutions in all test runs (see Chapter 5 and Appendix II). The optimal solution was not known for the test problem published in El-Sayed *et. al.* (1989). For this test problem, the proposed TS algorithm found a solution that was around 25% better than the best known solution. The solution obtained for the test problem published in Schniederjans and Santhanam (1989) was also better than the known solution (see Appendix II). Due to its problem-independent nature, the proposed algorithm is also a good candidate for the simulation optimisation applications. The application of the proposed TS algorithms to the simulation

optimisation problems has also been explained. A C++ computer program has also been developed during the study to implement the proposed TS algorithm.

The results of this research show that TS is a good candidate for solving MOO problems. The developed algorithms are a direct application of TS to MOO problems (Baykasoglu *et. al.,* 1999a-b). According to the detailed literature review, TS has not been directly applied to MOO optimisation problems before. As explained above (see also Chapters 4 and 5) the proposed TS algorithms have the ability to solve MOO problems.

b) Manufacturing Cell Formation

As pointed out by many researchers, cellular manufacturing systems are good candidates for improving the performance of job shops and implementing advanced manufacturing technologies, like CIM, FMS etc. (Wemmerlow and Johnson, 1997). However, formation of manufacturing cells is a key issue for a successful implementation of these systems. According to Shafer and Meredith (1990) the success of cellular manufacturing systems is mostly dependent on the success of their design. Zhou and Askin (1998) defined cell formation as the most important stage in designing a cellular manufacturing system. Due to its importance, cell formation received big attention in the literature. There are many procedures available in the literature for forming manufacturing cells and the corresponding part families. However, as discussed in Chapter 2 and Chapter 6, these procedures have several disadvantages in modelling and solving cell formation problems. For example, the majority of these techniques optimise only a single criterion although cell formation in practice must also be a multiple objective optimisation problem. Another

disadvantage is that they cannot offer mechanisms that can effectively evaluate alternative routes, machines etc. Also, they don't take into account machine capacities, part processing times, demand, processing sequences etc. A new cell formation technique is proposed in this study in order to overcome some of the disadvantages of the existing cell formation procedures. The proposed technique has the following distinguishable features:

- Many cell formation methodologies create part and machine cells in a sequential manner. In contrast with these methodologies, the proposed technique creates manufacturing cells and corresponding part families simultaneously. As mentioned in Chapter 2-Section 2.5, the simultaneous cell formation approach is advantageous.
- As mentioned above (see also Chapters 2 and 6) the majority of cell formation methodologies are based on single objective optimisation and do not consider several other important objectives and constraints. The cell formation problem is a multiple-objective decision making problem and has some constraints. In the proposed technique, the cell formation problem is modelled as a multiple objective optimisation problem. Part similarities based on their operation requirements and operation sequences, load balance between cells, flexibility of cells (in terms of maximising their capability scope), inter-cell movement and cell size constraints have been considered (see also Chapter 6). The proposed model uses Resource Elements (RE) to define part processing requirements and machine capabilities (REs have been explained in Appendix III). REs can define both part processing requirements and machine capabilities in the same language. Therefore, the proposed model can evaluate all possible process plans while forming part-machine cells. Knowledge of capability in cell formation increases

the chance of more realistic problem formulation. This increases the possibility of minimising extra capacity requirements, minimising inter-cell movements and maximising cell capacity utilisation in cell formation. The present model also proposes also a new formulation for capacity calculations, which does not require the actual machine based routes for the individual parts. The formulation uses transportation like equations. These features of the proposed model distinguish it from the existing cell formation models. Therefore manufacturing cell formation can be better achieved with the proposed model.

- The TS-based multiple-objective optimisation algorithm that was developed in Chapter 5 has been employed to solve the manufacturing cell formation model. Some problem-related enhancements have been also discussed in Chapter 6, in order to improve the capability of the TS algorithm. The proposed cell formation model is also superior to the one proposed by Kusiak (1987) and to some other cell formation models, in terms of its size, because it requires less variables, even though it is a more comprehensive formulation. This issue is especially important in solving large problems optimally, within a reasonable computation time.
- The proposed cell formation technique was compared with several other cell formation techniques available in the literature. The comparisons made with Kusiak (1987), and Seifoddini and Wolfe (1986) production flow analysis based methods showed that the proposed technique forms manufacturing cells with less machine duplication and better cell utilisation levels. Additionally, the proposed technique can determine what capability is required and how much capacity is needed, if there is a need for extra capacity. It recognises part-processing sequences while forming part families and several other important data, objectives

and constraints, as explained before (see also Chapter 6). These have been generally not taken into account in most other cell formation procedures.

- A computer program named MOCACEF 1.0 was also developed during this study. A FORTRAN 90 version of this program is available in Appendix IV.

c) Loading of Cellular Manufacturing Systems

As discussed in Chapters 2 and 7, part families constantly change under dynamic manufacturing environments, due to design and demand changes and the introduction of new part types. Under these circumstances the loading problem should be considered seriously in order to retain or improve the efficiency of cellular manufacturing systems. Cell loading is simply the determination of cell (or cells) to which a part should be assigned in order to satisfy the required system performance levels (see also Chapter 7). No serious attention has been given to the loading problem of CMS and the problem has not been totally addressed in the literature. During this research, this problem is studied in detail. A loading framework has been developed, for assigning parts to cells and scheduling manufacturing cells simultaneously. A *parts list* that contains machine-independent abstract process plans based on REs is the main input in the proposed loading system. Using these abstract process plans in cell loading enables the utilisation of alternative machines for part processing and results in seamless integration of process planning with loading. Parts were assigned to manufacturing cells based on their RE requirements and scheduled in each cell with a RE-based simulation-scheduling system (Gindy and Saad, 1996). This system can dynamically match part processing-requirements with the suitable machines, by considering the current status of the cellular manufacturing system.

A multiple objective simulation optimisation approach was proposed for the formal modelling of the loading problem. The model is hybrid in nature, i.e. some objectives were defined analytically others were obtained from a parametric simulation model. The model was implemented using the SIMAN simulation language and C++. Based on the literature review of this thesis no equivalent approach for solving the loading problems of CMS has been proposed before (see Chapter-2). Many characteristics of CMS cannot be formulated analytically. Therefore simulation must be used for modelling these characteristics (i.e. part arrivals, queuing times etc.). But optimisation cannot be achieved using only simulation. For this reason simulation must be integrated with the TS algorithm which optimises the loading model. Some problem-related enhancements were also discussed and explained in Chapter 7, in order to improve the capability of the TS algorithm for solving the proposed loading model. In this way it is possible to obtain realistic results for implementation.

The characteristics of the loading model have also been compared to the Greene and Sadowski (1986) loading model in Chapter 7. Their model is the only formal model available in the literature for loading of CMS. Unlike the loading model proposed in this thesis, the Greene and Sadowski (1986) model makes several assumptions which limit the model capabilities[1]. Consequently, it is less realistic than the TS-based model. As they concluded, their model can only be applied to very small problems, due to large number of variables necessary for modelling. The present TS-based model does not have these limitations (see Chapter 7). As was also mentioned in

[1] Some of these assumptions are: 'a part can only be assigned to one cell (no possibility of cell interaction) and there is at least one such cell for all parts', 'no cell has more than one machine of any type', 'cells do not have to have a machine of every type', 'part routes are machine based and fixed', 'a part type can visit any machine a maximum of once', additionally dynamic characteristics like part arrivals, queue lengths, etc. were not considered.

Chapter 7, research on CMS has been generally concentrated on the cell formation problem, but the cell-loading problem has not received much attention. Therefore, this current research is useful in filling this gap.

The results obtained using the proposed loading mechanism have also proven that it is possible to improve and/or sustain the performance of CMS facing changing production requirements. Implementation of such mechanisms is also useful while making decisions about reconfiguration of manufacturing cells, because they can assess the performance of CMS.

d) Reconfiguration of Cellular Manufacturing Systems

As discussed in previous chapters (1,2,3 and 8) in some production periods it is necessary to consider the reconfiguration of cellular manufacturing systems (CMS), due to unsatisfactory performance levels (high tardiness, low throughput etc.) It was also observed that, even though the reconfiguration of functionally divided job shops received some attention in the literature (Lacksonen and Enscore, 1993, Rosenbaltt, 1986), reconfiguration of CMS did not receive enough attention. However, the sensitivity of CMS to changing production requirements is very well known and several papers have been published on this issue (Sasani, 1990, Seifoddini and Djassemi, 1996,1997). As discussed in Chapter 2-Section 2.7.3, reconfiguration of CMS is a very complex problem and the reconfiguration strategies applied for functional job-shops are not totally adequate for CMS.

The virtual cell concept of McLean *et. al.* (1982) offers a practical strategy to reconfigure CMS. The reconfiguration approach proposed in this thesis was inspired

from their virtual cell concept. As discussed in Chapter 8, virtual cells have also been studied by Drolet and her colleagues (Drolet *et. al.*,1989,1990). But there are distinct differences between their implementation of virtual cells and the present approach. In their implementation, virtual cells were created continuously when a job order arrived. Based on the machine requirements of arriving jobs, first candidate machines were determined then virtual cells were generated via minimising the material travelling distance. There is no explicit decision made about reconfiguration in their application, i.e. reconfiguration is a natural process and happens continuously when a job order arrives. Additionally, in their implementation the possibility of using other performance measures for creating virtual cells was not considered. For instance, if multiple jobs are required to be processed simultaneously on the shop floor, the minimisation of travel distance for each job separately might not result in a good shop performance (Chatterjee, 1992). Capabilities of machines (i.e. alternative machines for part processing) were not considered while forming the virtual cells even though this might be beneficial. There is no mechanism to avoid machine sharing in their implementation. Machine sharing increases the complexity of control (Kusiak, 1990). They also concluded that their way of implementing virtual cells is only suitable for highly automated factories (Drolet, 1990) (see also Chapter 8).

In the present approach, the reconfiguration is also considered as an ongoing activity. However, virtual cell creation is explicitly decided in discrete time periods (i.e. at the beginning of each loading period), via interconnecting and integrating process planning, loading and cell formation activities. Virtual cell formation is considered as an objective driven activity. In other words, it is decided when performance measures

(utilisation, tardiness etc.) obtained from the loading system (Chapter 7) are found to be unsatisfactory.

A framework was proposed to guide and achieve reconfiguration activities based on the above logic, as explained in Chapter 3. In the present approach, Resource Elements (Appendix III) were used to define part processing requirements and capabilities of machines. Alternative machines for part processing were considered and machine sharing was avoided. A hybrid MOO model based on pre-emptive goal programming was proposed for generating virtual cell configurations (Chapter 8). The multiple-objective TS algorithm developed in Chapter 5 was applied to find the solution of the reconfiguration model. Problem-related enhancements to the TS algorithm have also been presented in Chapter 8.

The proposed framework can be used for the reconfiguration of traditional, as well as modern CMS. Moreover, test studies described in Chapter 8 and Appendix VII have shown that is possible to improve the performance of CMSs facing changing production requirements by reconfiguring them using virtual cells.

9.4 SUGGESTIONS FOR FUTURE WORK

Future work that can be considered as the modification of the proposed procedures and the framework developed in this study can be extended to the following areas:

a) Multiple Objective Optimisation

- On many occasions it may not be possible to set precise goals in a pre-emptive goal programming model. Additionally, the user can allow violation of some hard constraints in order to increase the flexibility of solving the mathematical model. These kinds of issues are generally modelled as fuzzy mathematical programming models. The proposed TS algorithm was applied to solve crisp MOO problems. However, the present algorithm can also be applied to solve fuzzy MOO problems with some modifications. This requires further research.
- The pre-emptive goal programming models developed in this thesis (i.e. cell formation model, loading model and reconfiguration model) can also be formulated as fuzzy goal programs and solved with the fuzzy version of the proposed TS algorithm. This may results in a better and more realistic problem representation. However there is a need to develop algorithms for their solution. As pointed out above, the present TS algorithm can be extended to solve the fuzzy version of these models.
- More computational work is necessary in order to investigate the effects of TS parameters on the solution quality. Determination of these parameters is also necessary in order to reduce computational time requirements and improve solution quality.
- In this research, quality of the solutions obtained from the multiple-objective TS algorithms are judged by comparing them with the known solutions of the test problems. However more research work is necessary in order to determine the optimality of solutions generated by the proposed TS algorithms. One possible way of doing this might be the determination of the lower and upper boundaries of the test problems where possible.

b) Simulation Optimisation

- In the present work a parametric simulation optimisation model was developed to solve the loading and reconfiguration models of this thesis. However, there is a need for general-purpose parametric simulation optimisation tools that can be used for various manufacturing applications, like the ones studied in this thesis. Further research is needed to develop general-purpose parametric simulation models that can be combined with optimisation models. This might be achieved by developing general-purpose systems that can automatically generate simulation models based on the data provided by the optimisation routines. Object-oriented or classical relational data base systems can be used for this purpose. Afterwards, each model can be evaluated with the optimisation system. Development of such systems is beyond the scope of this research. However, these systems can enable the designer to create better cell designs and make better reconfiguration decisions. Extensive research is required, as the topic has been scarcely addressed in the past.

c) Cell formation

The proposed cell formation model can be extended in a number of ways:

- In the present model, the flexibility of cells is defined in terms of number of different Resource Elements (RE) contained in cells. This flexibility increases the chance of accommodating future product mixes without increasing the inter-cell part movements. Other flexibility criteria like maximising the repetition of each RE in each formed cell can also be considered. This indicates the routing flexibility in each cell. However, incorporation of several flexibility types into the cell formation model is not easy and straightforward because, they might be conflicting (e.g. increasing number of different types of RE in each cell might

reduce the possibility of increasing repetition of each RE). One possible way of dealing with this kind of situations might be the investigation of the amount of flexibility needed to achieve the desired operational performance of the CMS.

- In the present cell formation model only machine resources are considered. Non-machine resources like humans and robots may also be considered in the initial cell formation step even though these resources are generally assigned to cells after determining the part-machine clusters. Their consideration in the initial cell formation stage might be advantageous even though it requires the determination of relations between various resources (e.g. relations between robot, machine and part). This might result in a very complicated formulation.
- The mathematical model can also be extended by including several other constraints and objective functions based on specific user requirements. For example, cell utilisation levels can be constrained by specifying upper and lower utilisation limits for cells.
- Extensive computational research is necessary in order to determine truly the computational time performance of the proposed cell formation algorithm and the optimality of the generated solutions.

d) Cell Loading

- The present cell-loading model demands too much computational power. This is also true for the reconfiguration model. This is mainly due to the number of simulation experiments required to solve the model. One possible way to reduce the computational time requirements might be achieved by replacing the simulation model with a well-trained neural network. It is also possible to develop a meta-model of the proposed loading (or reconfiguration) system to save the

computational time. The loading (or reconfiguration) model and a neural network may run in parallel and later the neural network may be replaced with the simulation model. This might significantly reduce the computational time requirements.

e) Cellular Manufacturing Reconfiguration

- In the present reconfiguration model, the virtual cell concept was used to reconfigure the CMS. The physical reconfiguration of the CMS was not considered. In order to deal with physical reconfiguration of a manufacturing system, several other issues should also be considered in problem modelling such as the movement cost of machines, replacing material handling links etc. All these add to the reconfiguration cost. Then reconfiguration decisions require resolution of the trade-off between performance improvement and cost of reconfiguration. However the literature is almost empty about the physical reconfiguration of CMS and development of such methods can be useful and this subject deserves further research effort.
- Intelligent procedures can be incorporated into the reconfiguration framework to forecast future demand, keep records of system performance measures in relation to various demand situations, and estimate the trend of system performance. These may help in determining the reconfiguration time.
- Extensive computational research is also necessary for loading and reconfiguration algorithms in order to determine their truly computational time performances and the optimality of the generated solutions. The effects of TS parameter setting on the solution quality need also be investigated. However, in these models the main computational burden is coming from the simulation

therefore the future work should concentration on the minimisation of the number of simulation experiments. One possible way may be the incorporation of an expert system which can make estimations on the performance of some feasible solutions generated by the TS algorithm which are believed (or obvious) to have poor performances and therefore disregard these solution vectors (i.e. do not simulate them and assign them unfavourable performance values in order the reduce their change of being selected in the search).

REFERENCES AND BIBLIOGRAPHY

Abedzadeh, M., O'Brien, C. (1996). The potential for virtual cells in a rapid reconfigurable environment. Proc. of the 1st Int. Conf. on Ind. Eng. App. and Practice, Dec. 4-6 1996, Texas, USA, pp.1120-1126.

Adil, G. K., Rajamani, D., Strong, D. (1996). "Cell formation considering alternate routings." *Int. J. Prod. Res.* **34**(5): 1361-1380.

Afentakis, P., Millen, R., Solomon, M. M. (1990). "Dynamic layout strategies for flexible manufacturing systems." *Int. J. Prod. Res.* **28**(2): 311-323.

Akturk, M. S., Balkose, H. O. (1996). "Part-machine grouping using a multi-objective cluster analysis." *Int. J. Prod. Res.* **34**(8): 2299-2315.

Al-Fawzan, M. A., Al-Sultan, K. S. (1998). A tabu search based algorithm for production planning when routing is flexible. 7th Annual Ind. Eng. research Conf., Banff, Aberta, Canada.

Al-qattan, I. (1990). "Designing flexible manufacturing cells using a branch and bound method." *Int. J. Prod. Res.* **28**(2): 325-336.

Albino, V., Garavelli, C. (1997). Performance evaluation of a cellular manufacturing system subject to demand variability. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Ang, C. L., Willey, P. C. T. (1984). "A comparative study of the performance of pure and hybrid group techology manufacturing systems using computer simulation techniques." *Int. J. Prod. Res.* **32**(2): 193-233.

Arzi, Y., Roll, Y. (1993). "Dispatching procedures for a flexible manufacturing cell in constant production circumstances." *Int. J. Prod. Res.* **13**: 35-51.

Bakir, M. A. (1996). A Hybrid Analytic/Simulation Modelling Approach to Production Planning. PhD Thesis. Nottingham, University of Nottingham.

Balakrishan, P. V., Jacob, V. S. (1996). "Genetic algorithms for product design." *Management Science* **42**: 1105-1117.

Ballakur, A., Steudel, H. J. (1987). "A within-cell utilisation heuristic for designing cellular manufacturing systems." *Int. J. Prod. Res.* **25**(5): 639-665.

Banks, J., Carson, J. S. (1986). Introduction to discrete-event simulation. Proc. of the 1986 Winter Simulation Conf.

Barad, M., Aharonson, Z. (1997). From dynamic layout to marketing flexibility. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Basu, A., Hyer, N., Shtub, A. (1989). "An expert system based approach to manufacturing cell design." *Int. J. Prod. Res.* **27**(10): 1637-1651.

Baykasoglu, A. (1995). Computer Aided Optimization of Cutting Conditions in Multicut Turning Operations. MsC Thesis, Gaziantep, University of Gaziantep, Turkey.

Baykasoglu, A., Saad, S. M., Gindy, N. N. Z. (1998-a). A Loading approach for cellular manufacturing systems. The 8th International Conference of FAIM'98, Portland State University, Oregon, USA pp. 215-226.

Baykasoglu, A., Gindy, N. N. Z., Saad, S. M. (1998-b). A framework for the reconfiguration of cellular manufacturing systems. The 2nd International Symposium on Intelligent Manufacturing Systems, August 6-7, 1998, Sakarya, Turkey, Sakarya University, pp 565-573.

Baykasoglu, A., Gindy, N. N. Z. (1999). "MOCACEF 1.0: Multiple objective capability based approach to form part-machine groups for cellular manufacturing applications." Accepted for publication in Int. J. Prod. Res.

Baykasoglu, A., Owen, S., Gindy, N. (1999-a). "A taboo search based approach to find the Pareto optimal set in multiple objective optimisation." Paper to be published in August issue of Engineering Optimization.

Baykasoglu, A., Owen, S., Gindy, N. (1999-b). "Solution of goal programming models using a basic taboo search algorithm." Paper accepted for publication in Journal of Operational Research Society.

Baykasoglu, A., Gindy, N. (1999-c). "A simulated annealing algorithm for concurrently forming independent part-machine cells via resource elements." Working Papers, University of Nottingham.

Benajaafar, S. (1995). Design of flexible layouts for manufacturing systems. IEEE Engineering Management Conference.

Bennage, W. A., Dhingra, A. K. (1995). "Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing." *Int. J. for Num. Meth. in Engineering* **38**: 2753-2773.

Bentley, P. J., Wakefield, J. P. (1998). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. Internet Publication: http://members.aol.com/docbentley/dispaper.htm.

Biegel, J. E., Davern, J. J. (1990). "Genetic algorithms and job shop scheduling." *Comput. Ind. Engng.* **19**: 81-91.

Biles, W. E. (1977). Optimization of multiple objective computer simulations: a nonlinear goal programming approach. 9th Annual Conf. American Instute for Decision Sciences.

Blackstone, J. H. (1982). "A state-of-art survey of dispatching rules for manufacturing job shop operations." *Int. J. Prod. Res.* **20**: 27-45.

Bland, J. A., Dawson, G. P. (1991). "Tabu search and design optimisation." *Computer Aided Design* **23**: 195-201.

Bley, H., Wuttke, C. C. (1997). "Distributed simulation applied to production systems." *Annals of the CIRP*, **46**: 361-364.

Boctor, F. F. (1991). "A linear formulation of the machine-part cell formation problem." *Int. J. Prod. Res.* **29**(2): 343-356.

Brah, S. A. (1996). "A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors." *Prod. Planning and Control* ,**7**(4): 362-373.

Burbidge, J. L. (1975). *The Introduction of Group Technology*, New York, Wiley.

Burbidge, J. L. (1987). Group technology in Yugoslavia. *Production Engineer*: 11-19.

Carvalho, J. D. A., Gindy, N. N. Z. (1995). Integration of process planning and scheduling using resource elements. Proc. Of the IEEE/ECCA/IFIP Int. Conf. On Architectures and Design methods for balances Automation Systems, Lisbon-Portugal. pp.203-210.

Chan, T. S., Pak, H. A. (1986). "Modelling of a controller for a flexible manufacturing cell." *Proc. Instn. Mech. Engrs.* **200**: 219-227.

Chang, T. C., Wysk, R. A., Wang, H. P. (1991). *Computer-Aided Manufacturing*, Prentice Hall.

Charnes, A., Cooper, W. W. (1961). *Management Models and Industrial Applications of Linear Programming*, John Wiley & Sons, Inc.

Charnes, J. M. (1993). Statistical analysis of output processes. Proc. of the 1993 Winter Simulation Conf.

Chatterjee, S. (1992). Resourcing in dynamic manufacturing. Proceedings of ASME, Concurrent Engineering. v. 59, pp. 361-374.

Chen, Y.-C., and Askin, R. G. (1990). "A multiobjective evaluation of flexible manufacturing system loading heuristics." *Int. J. of Prod. Res.* **28**(5): 895-911.

Chen, I. J., Chung, C. (1991). "Effects of loading and routing decisions on performance of flexible manufacturing systems." *Int. J. of Prod. Res.* **29**(11): 2209-2225.

Chen, W. H., Srivastava, B. (1994). "Simulated annealing procedures for forming machine cells in group technology." *European J. of Operational Research* **75**: 100-111.

Chen, C. L., Nepalli, R. V., Aljaber, N. (1996). "Genetic algorithms applied to the continuous flow shop problem." *Comput. Ind. Engng.* **30**(4): 919-929.

Chen, M. C., Tsai, D. M. (1996). "A simulated annealing approach for optimization of multi-pass turning operations." *Int. J. Prod. Res.* **34**(10): 2803-2825.

Chen, H. K., Chou, H. W. (1996). "Solving multiobjective linear programming problems-a generic approach." *Fuzzy Sets and Systems* **82**: 35-38.

Cheng, C., Goh, C., Lee, A. (1995). "A two-stage procedure for designing a group technology system." *Int. J. Opr. & Prod. Management* **15**(6): 41-50.

Cheng, R., Gen, M., Tsujimura, Y. (1996). "A tutorial survey of job-shop scheduling problems using genetic algorithms-1 representation." *Comput. Ind. Engng.* **30**(4): 983-997.

Cheng, C. H., Madan, M. S., Motwani, J. (1996). "Knapsack-based algorithms for designing cellular manufacturing systems." *J. of the Operational Research Society* **47**: 1468-1476.

Chipperfield, A. J., Fleming, P. J., Foncesa, C. M. (1994). A genetic algorithm toolbox for MATLAB. Proc. Int. Conf. Systems Engineering, Coventry, U.K.

Chipperfield, A., Fleming, P. (1996). "Multiobjective gas turbine engine controller design using genetic algorithms." *IEEE Trans. on Ind. Electronics* **43**(5): 583-587.

Choobineh, F. (1984). "Optimum loading for GT/MRP manufacturing systems." *Comput. Indus. Engng.* **8**(3-4): 197-206.

Choobineh, F., Burgman, T. (1984). "Transmission line route selection: an application of k-shortest paths and goal programming." *IEEE Trans. on Power and Systems* **PAS-103**: 3253-3259.

Choobineh, F. (1988). "A framework for the design of cellular manufacturing systems." *Int. J. Prod. Res.* **26**(7): 1161-1172.

Chow, W. S., Kusiak, A. (1988). Cluster analysis for group technology. *Industrial Engineering*: 70-72.

Chryssolouris, G., Chang, S. (1985). "An integrated approach to integrate process planning and scheduling" *Annals of the CIRP* **34**(1):413-417.

Chu, C. (1993). "Manufacturing cell formation by competitive learning." *Int. J. Prod. Res.* **31**(4): 829-843.

Clayton, E. R., Weber, W. E., Taylor, B. W. (1982). "A goal programming approach to the optimisation of multiresponse simulation models." *IIE Trans.* **14**(4): 282-287.

Co, H. C., Araar, A. (1988). "Configuring cellular manufacturing systems." *Int. J. Prod. Res.* **26**(9): 1511-1522.

Connolly, D. (1992). "General purpose simulated annealing." *J. Opl. Res. Soc.* **43**(5): 495-505.

Crookall, J. R., Lee, L. C. (1977). "Computer aided performance analysis and design of cellular manufacturing systems." *CIRP, Journal of Manufacturing Systems* **6**: 177-195.

Daellenbach, H. G., George, J. A., McNickle, D. C. (1983). *Introduction to Operations Research Techniques*, Allyn and Bacon Inc.

Dahel, N. E., Smith, S. B. (1993). "Designing flexibility into cellular manufacturing systems." *Int. J. Prod. Res.* **31**(4): 933-945.

De, P. K., Acharya, D., Sahu, K. C. (1982). "A Change-constraint goal programming model for capital budgeting." *Journal of the Operational Research Society* **33**: 635-638.

Dereli, T., Baykasoglu, A., Gindy, N.N.Z., Filiz, I.H. (1998). Determination of optimal turret-index positions of cutting tools by using genetic algorithms. IMS-98, 2nd Int. Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey.

Dhingra, A. K., Lee, B. H. (1994). "A genetic algorithm approach to single and multiobjective structural optimization with discrete-continuous variables." *Int. J. for Num. Meth. in Engineering* **37**: 4059-4080.

Doulgeri, Z., Hibberd, R. D., Husband, T. M. (1987). "The scheduling of flexible manufacturing systems." *Annals of the CIRP* **36**: 343-346.

Driscol, J., sawyer, J. H. F. (1985). "A computer model for investigating the relayout of batch production areas." *Int. J. Prod. Res.* **23**(4): 783-794.

Drolet, J. R., Moodie, C. L., Montreuil, B. (1989). "Scheduling factories of the future." *J. of Mechanical Working Technology* **20**: 183-194.

Drolet, J. R., Montreuil, B., Moodie, C. L. (1990). Virtual cellular manufacturing layout planning. Int. Industrial Engineering Conf. Proc.

Drolet, J. R., Montreuil, B., Moodie, C. L. (1995). "Scheduling framework for virtual cellular manufacturing" *Int. J. of manufacturing System Design* **1**:351-365.

Dyer, J. (1972). "Interactive goal programming." *Management Science* **19**: 62-70.

El-Sayed, M. E. M., Ridgely, B. J., Sandgren, E. (1989). "Nonlinear structural optimization using goal programming." *Computers and Structures* **32**: 69-73.

Elmaraghy, H. A., Gu, P. (1989). "Feature based expert parts assignment in cellular manufacturing." *Journal of Manufacturing Systems* **8**: 139-152.

Elperin, T., Weissberg, I., Zahavi, E. (1990). "Machine design optimization by the monte carlo annealing method." *Eng. Opt.* **15**: 193-203.

Ettl, M., Schwehm, M. (1994). A design methodology for kanban-controlled production lines using queueing networks and genetic algorithms. Proc. Int. Conf. Systems Engineering, Coventry, U. K.

Fang, H. L., Ross, P., Corne, D. (1995). A promising algorithm approach to job shop scheduling, rescheduling, and open shop scheduling problems. Int. Conf. on GA in Eng. Sys.: Innovations and App. (GALESIA'95), U. K.

Ferrel, W., Chester, H. M., Edward, C. R. (1975). Optimisation techniques for computerized simulation models. Report, Los Angelas, California.

Filiz, I. H., Sonmez, A. I., Baykasoglu, A., Dereli, T. (1996). "Computer aided optimisation of cutting conditions in CNC turning operations (in Turkish)." *Makina Tasarim ve Imalat Dergisi* **3**(2): 66-78.

Fogel, D. B. (1995). "A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems." *Simulation* **64**: 399-406.

Foncesa, C. M., Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. Proc. 5th Int. Conf. on GA.

Fujimoto, H., Lian-yi, C., Tanigawa, Y., Iwahashi, K. (1995). FMS scheduling by hybrid approaches using genetic algorithm and simulation. GA in Eng. Systems: Innovations and Applications.

Galbraith, L., Standridge, C. R. (1994). "Analysis in manufacturing systems simulation: A case study." *Simulation* **63**: 368-375.

Gallagher, C. C., Knight, W. A. (1986). Group Technology Production Methods in Manufacture, Ellis Horwood Limited.

Gangadharan, R., Rajendran, C. (1994). "A simulated annealing heuristic for scheduling in a flowshop with bicriteria." *Comput. Ind. Engng.* **27**(1-4): 473-476.

Garcia-Diaz, A., Hogg, G. L., Tari, F. G. (1981). "Combining simulation and optimisation to solve the multimachine interference problem." *Simulation* : 193-201.

Gen, M., Cheng, R. (1997). Genetic Algorithms and Engineering Design, John Wiley and Sons, Inc.

Gindy, N. N. Z. (1989). "A hierarchical structure for form features." *Int. J. Prod. Res.* **27**(12): 2089-2103.

Gindy, N. Z., Ratchev, T. M. (1992). Process Capability Modelling for CIM Applications. Internal Report, Nottingham, University of Nottingham.

Gindy, N. N. Z., Huang, X., Ratchev, T. M. (1993). "Feature-based component model for computer-aided process planning systems." *Int. J. Computer Integrated Manufacturing* **6**: 20-26.

Gindy, N. N. Z., Ratchev, T. M., Case, K. (1995). "Component grouping for GT applications-a fuzzy clustering approach with validity measure." *Int. J. Prod. Res.* **33**(9): 2493-2509.

Gindy, N. N. Z., Ratchev, T. M., Case, K. (1996). "Component grouping for cell formation using resource elements." *Int. J. Prod. Res.* **34**: 727-752.

Gindy, N. N., Saad, S. M. (1996). Resource-based scheduling in virtual manufacturing environments. The proceedings of the 12th Int. conf. on CAD/CAM Robotics and Factories of the Future, 14-16, August, 1996, London, UK. pp.710-715.

Gindy, N. N., Saad, S. M. (1997). Coping with disturbances a performance evaluation approach in virtual manufacturing environments. Proc. of the 7th Int. FAIM Conf., 25-25, June 1997, Middlesbrough, U.K. pp.1-12.

Gindy, N. N. Z., (1997). Resource-based representation of manufacturing environments: The virtual factory. Discussion document, University of Nottingham.

Gindy, N. N., Saad, S. M. (1998). "Flexibility and responsiveness of machining environments." *Integrated Manf. Systems* 9(4): 218-227.

Glover, F. (1986). "Future paths for integer programming and linkage to artificial intelligence." *Computers and Operations Research* **13**: 533-549.

Glover, F. (1990). "Tabu search: a tutorial." *Interfaces* **20**(4): 74-94.

Glover, F. (1993). "A user's guide to tabu search." *Annals of Oper. Research*.

Goffe, W. L., Ferrier, G. D., Rogers, J. (1994). "Global optimisation of statistical functions with simulated annealing." *Journal of Econometrics* **60**: 65-99.

Gokcen, H., Erel, E. (1997). "A goal programming approach to mixed-model assembly line balancing problem." *Int. J. Prod. Economics* **48**: 177-185.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley.

Gonzalez, B., Torres, M., Moreno, J. A. (1995). A hybrid genetic algorithm approach for the no-wait flowshop scheduling problem. IEE: Gen. Alg. in Eng. Sys.: Innovations and App.

Greene, T. J., Sadowski, R. P. (1980). Loading cellularly divided group technology manufacturing system. IE Conf. Proc., Fall 1980, pp.190-195.

Greene, T. J., Sadowski, R. P. (1983). "Cellular manufacturing control." *J. of Manufacturing Systems* **2**(2): 137-144.

Greene, T. J., Sadowski, R. P. (1986). "A mixed integer program for loading and scheduling flexible manufacturing cells." *European Journal of Operational Research* **24**: 379-386.

Gunasingh, K. R., Lashkari, R. S. (1989). "The cell formation problem in cellular manufacturing systems-a sequential modelling approach." *Computers Ind. Engng* **16**(4): 469-476.

Gupta, T., Seifoddini, H. (1990). "Production data based similarity coefficient for machine-component grouping decisions in the design of cellular manufacturing system." *Int. J. Prod. Res.* **28**(7): 1247-1269.

Gupta, Y., Gupta, M., Kumar, A., Sundaram, C. (1996). "A genetic algorithm -based approach to cell composition and layout design problems." *Int. J. Prod. Res.* **34**(2): 447-482.

Haddock, J., Mittenthal, J. (1992). "Simulation optimisation using simulated annealing." *Comput. Ind. Engng.* **22**(4): 387-395.

Hadley, S. W. (1996). " Finding part machine families using graph partitioning techniques." *Int. J. Prod. Res.* **34**(7): 1821-1839.

Han, C., Ham, I. (1989). "Multi-objective cluster analysis for part family formations." *Journal of Manufacturing Systems* **5**(4): 223-229.

Harhalakis, G., Ioannou, G., Minis, I., Nagi, R. (1994). "Manufacturing cell formation under random product demand." *Int. J. Prod. Res.* **32**(1): 47-64.

Hartl, D. L., Freifelder, D., Snyder, L. A. (1988). *Basic Genetics*, Jones and Bartlett Publishers Inc.

Hertz, A., Werra, D. (1987). "Using tabu search techniques for graph colouring." *Computing* **29**: 345-351.

Hertz, A. (1991). "Tabu search for large scale time tabling problems." *European J. of Operational Research* **51**: 39-47.

Holland, J. H. (1992). "Genetic algorithms." *Scientific American*: 44-50.

Holthaus, O., Ziegler, H. (1997). "Improving job shop performance by co-ordinating dispatching rules." *Int. J. Prod. Res.* **35**(2): 539-549.

Homafair, A., Guan, S., Liepins, G. E. (1995). A new approach on the travelling salesman problem by genetic algorithms. Int. Conf. on GA in Eng. Sys.: Innovations and App.(GALESIA'95), U. K.

Homaifar, A., Qi, C. X., Lai, S. H. (1994). "Constrained optimization via genetic algorithms." *Simulation* **62**: 242-254.

Hon, K. K. B., Chi, H. (1994). "A new approach of group technology part families optimisation." *Annals of the CIRP* **43**(1): 425-428.

Horn, J., Nafpliotis, N., Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multi-objective optimization. Proc. of 1st IEEE ICEC.

Hsu, C. C., Yamada, S. I., Fujikawa, H., Shida, K. (1996). "A fuzzy self-tuning parallel genetic algorithm for optimization." *Comput. Ind. Engng.* **30**(4): 883-893.

Hsu, C., Su, C. (1998). "Multi-objective machine-component grouping in cellular manufacturing: a genetic algorithm." *Production Planning & Control* **9**(2): 155-166.

Huang, P. Y. (1984). "A comparative study of priority dispatching rules in a hybrid assembly job shop." *Int. J. Prod. Res.* **22**(3): 375-387.

Husbands, P., McIlhagga, M., Ives, R., Ed. (1995). Experiments with an ecosystems model for integrated production planning. Handbook of Evolutionary Computation, IOP Pub. Ltd. and Oxford University Press.

Hwang, H., Sun, J. (1996). "A genetic-algorithm-based heuristic for the GT cell formation problem." *Computers Ind. Engng* **30**(4): 941-955.

Ignizo, J. P. (1976). *Goal Programming and Extensions*, Heath, Lexington, Mass.

Ignizo, J. P. (1982). *Linear Programming in Single & Multiple Objective Systems*, Prentice Hall.

Ijiri, Y. (1965). *Management Goals and Accounting for Control*. Chicago, Rand-McNally.

Irani, S. A., Cohen, P. H., Cavalier, T. M. (1992). " Design of cellular manufacturing systems." *Trans. of the ASME, journal of Engineering for Industry* **114**: 352-361.

Irani, S. A. (1993). "Virtual manufacturing cells: Exploiting layout design and intercell flows for the machine sharing problem." *Int. J. Prod. Res.* **31**: 791-810.

Irastorza, J. C., Deane, R. H. (1974). "A loading and balancing methodology for job shop control." *AIIE Transactions* **6**(4): 302-307.

Islam, A., Eksioglu, M. (1997). "A Taboo search approach for the single machine mean tardiness problem." *Journal of the Operational Research Society* **48**: 751-755.

Iwata, K., Onosato, M., Koike, M. (1994). "Random manufacturing system: A new concept of manufacturing systems for production to order." *Annals of the CIRP* **43**: 379-383.

Jain, A. K., Elmaraghy, H. A. (1997). "Production scheduling/rescheduling in flexible manufacturing." *Int. J. Prod. Res.* **35**(1): 281-309.

Janikow, C. Z., St.Clair, D. (1995). Genetic algorithms. *IEEE Potentials*: 31-35.

Jinxing, X. (1995). An application of genetic algorithms for general dynamic lot-sizing problems. IEE: Gen. Alg. in Eng. Sys.: Innovations and App.

Joines, J. A., Culbreth, C. T., King, R. E. (1996). "Manufacturing cell design: an integer programming model employing genetic algorithms." *IEE Transactions* **28**: 69-85.

Jones, M. S., Russel, R. S. (1990). "Multiple performance measures in the selection of a sequencing rule." *Int. J. Opr. & Prod. Mang.* **10**: 29-41.

Joshi, B., Morris, D., White, N., Unal, R. (1996). Optimisation of operations resources via discrete evenet simulation modelling. 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimisation.

Kaandorp, J. A. (1994). Fractal modelling growth and form in biology, Springer-Verlag.

Kamal, S., Burke, L. I. (1996). "FACT:A new neural network-based clustering algorithm for group technology." *Int. J. Prod. Res.* **34**(4): 919-946.

Kamrani, A. K., Hubbard, K., Parsaei, H. R., Leep, H. R. (1998). "Simulation based methodology for machine cell design." *Computers Ind. Engng* **34**(1): 173-188.

Kannan, V. R., Ghosh, S. (1996). "Cellular manufacturing using virtual cells." *Int. J. Opr. & Prod. Management* **16**: 99-112.

Kannan, V. R. (1997). "A simulation analysis of the impact of family configuration on virtual cellular manufacturing." Production Planning & Control **8**(1): 14-24.

Kaparthi, S., Suresh, N. C. (1992). "Machine-component cell formation in group technology: A neural network approach." *Int. J. Prod. Res.* **30**: 1353-1367.

Kato, H., Hori, Y., Morikawa, K., Takahashi, K., Nakamura, N. (1997). Minimizing the mean tardiness of the job shop with tabu search. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Kayacan, M. C., Filiz, I.H., Sonmez, A.I., Baykasoglu, A., Dereli, T., (1996). "OPPS-ROT: An optimised process planning system for rotational parts." *Computers in Industry* **32**: 181-195.

Kazerooni, M., Luong, L. H. S., Abhary, K., Chan, F. T. S., Pun, F. (1996). An integrated method for cell layout problem using genetic algorithms. The proceedings of the 12th International conference on CAD/CAM Robotics and Factories of the Future, London, UK.

Kelton, W. D. (1994). Analysis of output data. Proc. of the 1994 Winter Simulation Conf.

Kerr, D. C., Balakrishnan, J. (1996). "Manufacturing cell formation using spreadsheets." *Int. J. Opr. & Prod. Management* **16**(9): 60-73.

Ketcham, M. G., Watt, J. (1989). "Parametric simulation for flexible assembly systems." *J. of Manufacturing Systems* **8**(2): 115-124.

Khator, S., Moodie, C. (1979). "A machine loading procedure which considers part complexity and machine capability." *Int. J. Prod. Res.* **17**(1): 33-44.

Khator, S. K., Irani, S. A. (1987). "Cell formation in GT: A new approach." *Computers Ind. Engng* **12**(2): 131-142.

Kidwell, M. D. (1993). Using genetic algorithms to schedule distributed tasks on a bus-based system. Proc. 5th Int. Conf. on GA.

Kim, Y.-D. (1993). "A study on surrogate objectives for loading a certain type of flexible manufacturing systems." *Int. J. of Prod. Res.* **31**(2): 381-392.

Kim, H., Nara, K., Gen, M. (1994). "A method for maintenance scheduling using GA combined with SA." *Comput. Ind. Engng.* **27**(1-4): 477-480.

Kirkpatrick, S., Gelatt, Jr., C. D., Vecchi, M. P. (1983). "Optimisation by simulated annealing." *Science* **220**: 671-680.

Kochikar, V., Narendran, T. T. (1998). "Logical cell formation in FMS, using flexibility based criteria." *The Int. J. of Flexible Manuf. Sys.* **10**: 161-181.

Kornbluth, J. S. H. (1986). " Engineering design: applications of goal programming and multiple objective linear and geometric programming." *Int. J. Prod. Research* **24**: 945-953.

Kumar, N. S. H., Srinivasan, G. (1996). "A genetic algorithm for job shop scheduling- A case study." *Computers in Industry* **31**: 155-160.

Kusiak, A. (1987). "The generalised group technology concept." *Int. J. Prod. Res.* **25**(4): 561-569.

Kusiak, A., Chow, W. S. (1988). "Decomposition of manufacturing systems." *IEEE J. of Robotics and Automation* **4**(5): 457-471.

Kusiak, A. (1990). Intelligent Manufacturing Systems, Prentice Hall.

Kusiak, A. (1995). Loading models in flexible manufacturing systems. In. Flexible Manufacturing Systems: Recent Developments. Elsevier Science B.V.

Lacksonen, T. A., Enscore, E. E. (1993). "Qadratic assignment algorithms for the dynamic layout problem." *Int. J. Prod. Res.* **31**(3): 503-517.

Laguna, M., Barnes, J. W., Glover, F. (1991). "Tabu search methods for a single machine scheduling problem." *Journal of Int. Manufacturing* **2**: 63-74.

Lee, S., Wang, H. P. (1990). "Controlling a robotic cell by dynamic rule despatching-a simulation study." *Int. J. Opr.& Prod. Mang.* **10**: 51-64.

Lee, Y. H., Iwata, K. (1991). "Part ordering through simulation-optimization in an FMS." *Int. J. Prod. Res.* **29**: 1309-1323.

Lee, M., Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. Proc. of the 5th Int. Conf. on GA, Urbana-Champaing.

Lee, I., Sikora, R., Shaw, M. J. (1993). Joint lot sizing and sequencing with genetic algorithms for scheduling: evolving the chromosome structure. Proc. of the 5th Int. Conf. on GA, Urbana-Champaing.

Lee, H., Garcia-diaz, A. (1993). "A network flow approach to solve clustering problems in group technology." *Int. J. Prod. Res.* **31**(3): 603-612.

Lee, J. (1996). "Composite dispatching rules for multiple-vehicle AGV systems." *Simulation* **66**(2): 121-130.

Leeuwen, E. H., Norrie, D. (1997). Holons and holarchies. *Manufacturing Engineering*: 86-88.

Liang, M., Dutta, S. P. (1990). "A mixed-integer programming approach to the machine loading and process planning problem in a process layout environment." *Int. J. of Prod. Res.* **28**(8): 1471-1484.

Liepins, G. E., Hillard, M. R. (1987). Greedy genetics. Proc. of the 2nd Int. Conf. on GA.

Liepins, G. E., Hillard, M. R. (1989). "Genetic algorithms: foundations and applications." *Annals of the O. R.* **21**: 31-58.

Liles, D. H., Huff, B. L. (1990). "A computer based production scheduling architecture suitable for driving a reconfigurable manufacturing systems" *Computers Ind. Engng.* **19**:1-5.

Lin, C. Y., Hajela, P. (1992). "Genetic algorithms in optimisation problems with discrete and integer design variables." *Eng. Opt.* **19**: 309-327.

Lin, F. T., Kao, C. Y., Hsu, C. C. (1993). "Applying the genetic approach to simulated annealing in solving some NP-hard problems." *IEEE Trans. on Systems, Man and Cybernetics* **23**(6): 1752-1767.

LINDO Systems Inc. (1996). *LINDO user's manual*. LINDO Systems Inc.

Liu, C. M., Kao, R. L., Wang, A. H. (1994). "Solving location-allocation problems with rectilinear distances by simulated annealing." *J. Opl. Res. Soc.* **45**(11): 1304-1315.

Liu, J. (1997). Controlling the parameters in simulated annealing for flowshop scheduling. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Logar, B., Peklenik, J. (1991). "Feature-based part database design and automatic forming of part families." *Annals of the CIRP* **40**(1): 153-156.

Louis, S., Rawlins, G. J. E. (1993). Pareto optimality, GA-easiness and deception. Proc. 5th Int. Conf. on GA.

Luong, L. H. S. (1993). " A cellular similarity coefficient algorithm for design of manufacturing cells." *Int. J. Prod. Res.* **31**(8): 1757-1766.

Maher, M. L., Garza, A. G. S. (1996). The adaptation of structural system designs using genetic algorithms. Proc. of the Int. Conf. on Information Tech. in Civil and Structural Engineering Design Stock and Future Directions, Glasgow- Scotland.

Malasri, S., Martin, J. R., Medina, R. A. (1996). "Solving mathematical programming problems using genetic algorithms." *Computing in Civil Engineering*: 233-239.

Malhotra, R., Mellichamp, J. M. (1997). "Intelligent simulation code generator: a relational database management approach." *J. of Intelligent Manufacturing* **8**: 125-136.

Man, K. F., Tang, K. S., Kwog, S. (1996). "Genetic algorithms: Concept and applications." *IEEE Trans. on Ind. Electronics* **43**: 519-533.

Manz, E. M., Haddock, J., Mittenthal, J. (1989). Optimization of an automated manufacturing system simulation model using simulated annealing. Proc. of the 1989 Winter Simulation Conf.

Marcotte, S., Montreuil, B., Lefrancois, P. (1998). Holographic layout design: towards manufacturing agility. 7th Annual Industrial Engineering Research Conference, Banff, Aberta, Canada.

Marett, R., Wright, M. (1996). "A comparison of neighbourhood search techniques for multi-objective combinatorial problems." *Computers Ops. Res.* **23**(5): 465-483.

Markus, A., Vancza, K., Monosstori, L. (1996). "A market approach to holonic manufacturing." *Annals of the CIRP* **45**: 433-436.

Marsh, R. F., Meredith, J. R., McCutcheon, D. M. (1997). "The life cycle of manufacturing cells." *Int. J. of Oper. & Prod. Manag.* **17**(12): 1167-1182.

Maturana, F., Gu, P., Naumann, A., Norrie, D. H. (1997). "Object oriented job-shop scheduling using genetic algorithms." *Computers in Industry* **32**: 281-294.

McCarthy, I., Ridgway, K. (1995). "Implementation of focused manufacturing techniques in a hand tool company." *Proc. Instn. Mech. Engrs.* **209**: 237-243.

McLean, C. R., Bloom, H. M., Hopp, T. H. (1982). The virtual cell. Proc. Of 4th IFAC/IFIP Conf. on Information Control Problems in Manufacturing Technology, Gaithersburg, MD, October, 1986, pp. 207-215.

McMillan, J. C. (1975). Mathematical Programming, John Wiley & Sons.

Michalewicz, Z., Dasgupta, D., Riche, G. L., Schoenauer, M. (1996). "Evolutionary algorithms for constrained engineering problems." *Comput. Ind. Engng.* **30**(4): 851-870.

Michalewicz, Z. (1996). Genetic Algorithms+Data Structures=Evolution Programs, Springer.

Mizugaki, Y., Hao, M., Sakamoto, M., Makino, H. (1994). "Optimal tool selection based on genetic algorithm in a geometric cutting simulation." *Annals of the CIRP* **43**: 433-436.

Mohamed, Z. M. (1996). "A flexible approach to (re)configure flexible manufacturing cells." *European J. of Oper. Res.* **95**: 566-576.

Mollaghasemi, M., Evans, G. W. (1994). "Multicriteria design of manufacturing systems through simulation optimisation." *IEEE Trans. on Sys., Man, and Cyb.* **24**(9): 1407-1411.

Montazeri, M., Van Wassenhove, L. N. (1990). "Analysis of scheduling rules for an FMS." *Int. J. Prod. Res.* **28**: 785-802.

Montreuil, B., Venkatadri, U., Lefrancois, P. (1991). Holographic layout of manufacturing systems. 19th IIE Systems Integration Conference, Orlando, Florida, USA.

Montreuil, B., Drolet, J., Lefrancois, P. (1992). The design and management of virtual cellular manufacturing systems. American Production & Inventory Control Society Conf. Proc.

Moon, Y. B. (1990). "Forming part-machine families for cellular manufacturing: A neural network approach." *Int. J. Adv. Manuf. Tech.* **5**: 278-291.

Moon, Y. B., Chi, S. C. (1994). "Generalised part family formation using neural networks." *Journal of Manufacturing Systems* **11**(3): 149-159.

Moon, C. M., Kim, J., Gen, M. (1997). A genetic algorithm based approach for design of independent manufacturing cells. The 14th International Conf. on Prod. Res.

Moreira, D. A. (1998). Agents Research, http://java.icmsc.sc.usp.br/.

Morgan, S. (1994). Automatic generation of QUEST simulation models. Proc. of the DENEB Users Group Conf.

Murata, T., Ishibuchi, H. (1995). MOGA: multi-objective genetic algorithms. Proc. of 2nd IEEE-ICEC.

Murata, T., Ishibuchi, H., Tanaka, H. (a) (1996). "Multi-objective genetic algorithm and its applications to flow shop scheduling." *Computers In. Engng.* **30**(4): 957-968.

Murata, T., Ishibuchi, H., Tanaka, H. (b) (1996). "Genetic algorithms for flowshop scheduling problems." *Comput. Ind. Engng.* **30**(4): 1061-1071.

Murata, T., Ishibuchi, H. (1997). Performance of multi-objective genetic algorithms for flowshop scheduling problems. The 14th International Conference on Production Research, Osaka- Japan.

Myint, S., Taboocanon, M. T. (1994). "A multiple criteria approach to machine selection for flexible manufacturing system." *Int. J. Prod. Economics* **33**: 121-131.

Nelder, J. A. (1965). "A simplex method for function minimization." *Computer Journal* **7**: 308-313.

Nolan, R. L., Sovereign, M. G. (1972). "A recursive optimization and simulation approach to analysis with an application to transportation systems." *Management Science* **18**(12): 676-690.

Norman, V. B., Norman, T. A. (1991). Simulation and advanced manufacturing system design. Proc. of the 1986 Winter Simulation Conf.

Offodile, O. F. (1992). "Assignment model formulation of the machine cell formation problem in cellular manufacturing." *Int. J. Opr. & Prod. Management* **13**(10): 49-59.

Offodile, O. F., Grznar, J. (1997). "Part family formation for variety reduction in flexible manufacturing systems." *Int. J. Opr. & Prod. Management* **17**(3): 291-304.

Osyczka, A., Kundu, S. (1996). "A Modified distance method for multicriteria optimization using genetic algorithms." *Computers Ind. Engng.* **30**(4): 871-882.

Pearce, R., Cowley, P. H. (1996). "Use of fuzzy logic to describe constraints derived from engineering judgment in genetic algorithms." *IEEE Trans. on Industrial Electronics* **43**(5): 535-540.

Pegden, D., Ham, I. (1982). "Simulation of manufacturing systems using SIMAN." Annals of the CIRP **31**: 365-369.

Pegden, C. D., Shannon, R. E., Sadowski, R. P. (1995). Introduction to Simulation Using SIMAN, McGraw Hill.

Perotti, G., Tornincasa, S., Oberto, G. (1991). "Semantic techniques for representation and identification of part families." *Annals of the CIRP* **40**(1): 451-454.

Pinedo, M. (1995). Scheduling: Theory, Algorithms, and Systems, Prentice Hall.

Polajnar, A., Buchmeister, B., Leber, M. (1995). "Analysis of different transport solutions in the flexible manufacturing cell by using computer simulation." *Int. J. of Oper. & Prod. Management* **15**(6): 51-58.

Prakash, S., Shannon, R. E. (1993). "Development of a goal directed simulation environment for discrete part manufacturing systems." *Simulation* **61**(2): 102-115.

Punch, W. F., Goodman, E. D., Pei, M., Chia-Shun, L., Hovland, P., Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. Proc. 5th Int. Conf. on GA.

Purcheck, G. (1985). "Machine-component group formation: an heuristic method for flexible production cells and flexible manufacturing systems." *Int. J. Prod. Res.* **23**(5): 911-943.

Raghu, T. S., Rajendran, C. (1995). "Due-date setting methodologies based on simulated annealing- an experimental study in a real-life job shop." *Int. J. Prod. Res.* **33**(9): 2535-2554.

Rajamani, D., Singh, N., Aneja, Y. P. (1992). "A model for cell formation in manufacturing systems with sequence dependence." *Int. J. Prod. Res.* **30**(6): 1227-1235.

Ramanathan, R., Ganesh, L.S. (1995). "Energy alternatives for lighting in households: an evaluation using and integrated goal programming-AHP model." *Energy* **20**: 63-72.

Ramaswamy, S. E., Joshi, S. B. (1994). "A response surface approach to developing composite dispatching rules for flexible manufacturing systems." *Int. J. Prod. Res.* **32**(11): 2613-2629.

Rao, S. S., Dhingra, A. K., Miura, H. (1990). "Pareto-optimal solutions in helicopter design problems." *Engineering Optimization* **15**: 211-231.

Reeves, C. R. (1995). Modern Heuristic Techniques for Combinatorial Problems, McGraw Hill.

Reisman, A., Kumar, A., Motwani, J., Cheng, C. H. (1997). "Cellular manufacturing: A statistical review of the literature(1965-1995)." *Operations Research* **45**(4): 508-520.

Rheault, M., Drolet, J. C., Abdulnour, G. (1995). "Physically reconfigurable virtual cells: a dynamic model for a highly dynamic environment." *Comp. Ind. Engng* **29**(1-4): 221-225.

Ro, I., Kim, J. (1990). "Multi-criteria operational control rules in flexible manufacturing systems (FMSs)." *Int. J. Prod. Res.* **28**: 47-63.

Roach, A., Nagi, R. (1996). "A hybrid GA-SA algorithm for just in time scheduling of multi level assemblies." *Comput. Ind. Engng.* **30**(4): 1047-1060.

Rosenbaltt, M. J. (1986). "The dynamics of plant layout." *Management Science* **32**(1): 76-85.

Rutenbar, R. A. (1989). Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*. **1:** 19-26.

Saad, S. M. I. (1994). Design and Analysis of a Flexible Hybrid assembly System. PhD Thesis, Nottingham, University of Nottingham.

Saad, S. M., Baykasoglu, A., Gindy, N. (1998). "Integrated system for loading and scheduling in cellular manufacturing." Submitted to Prod. Planning and Control.

Sakawa, M., Kato, K., Mori, T. (1996). "Flexible scheduling in a machining centre through genetic algorithms." *Comput. Ind. Engng.* **30**(4): 931-940.

Sankaran, S. (1990). "Multiple objective decision making approach to cell formation: A goal programming model." *Mathl. Comput. Modelling* **13**(9): 71-82.

Sargent, R. G. (1994). Verification and validation of simulation models. Proc. of the 1994 Winter Simulation Conf.

Sarker, B. R., Balan, C. V. (1996). "Cell formation with operation times of jobs for even distribution of workloads." *Int. J. Prod. Res.* **34**(5): 1447-1468.

Sassani, F. (1990). "A simulation study on performance improvement of group technology cells." *Int. J. Prod. Res.* **28**: 293-300.

Satake, T., Morikawa, K., Takahashi, K., Nakamura, N. (1997). Simulated annealing approach for minimising the makespan of the general job-shop. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Sauer, W., Weigert, G., Hampel, D. (1997). An open optimisation system for controlling of manufacturing processes. Int. FAIM'97 Conf.

Schniederjans MJ, Kwak NK (1982). "An alternative solution method for goal programming problems: a tutorial"*The Journal of the Operational Research Society* **33**: 247-251.

Schniederjans MJ, Santhanam (1989). "A zero-one goal programming approach for the journal selection and cancellation problem" *Computers Opns. Res.* **6**: 557-565.

Schaffer, J. D. (1985). Multiple objective optimisation with vector evaluated genetic algorithms. Genetic Algorithms and Their Applications: Proc. of the 1st Int. Conf. on GA.

Schoenauer, M., Xanthakis, S. (1993). Constrained GA optimization. Proc. 5th Int. Conf. on GA.

Seifoddini, H., Wolfe, P. M. (1986). "Application of the similarity coefficient method in group technology." *IIE Transactions* **9**: 271-277.

Seifoddini, H., Djassemi, M. (1996). "Sensitivity analysis in cellular manufacturing system in the case of product mix variation." *Comp. Ind. Engng.* **31**: 163-167.

Seifoddini, H., Djassemi, M. (1997). "Determination of a flexibility range for cellular manufacturing systems under product mix variations." *Int. J. Prod. Res.* **35**(12): 3349-3366.

Selim, S. Z., Alsultan, K. (1991). "A simulated annealing algorithm for the clustering problem." *Pattern Recognition* **24**(10): 1003-1008.

Shafer, S. M., Meredith, J. R. (1990). "A comparison of selected manufacturing cell formation techniques." *Int. J. Prod. Res.* **28**(4): 661-673.

Shaw, K. J., Fleming, P. J. (1996). Initial study of multi-objective genetic algorithms for scheduling the production of chilled ready meals. 2nd Int. Mendel Conf. on GA.

Shimizu, M. (1991). "Application of an integrated modelling approach to design and analysis of manufacturing systems." *Adv. Manuf. Eng.* **3**: 3-17.

Siarry, P., Berthiau, G. (1997). "Fitting of Taboo search to optimise functions of continuous variables." *Int. J. for Numerical Methods in Engineering* **40**: 2449-2457.

Sikora, R. (1996). "A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line." *Comput. Ind. Engng.* **30**(4): 969-981.

Smith, D. E. (1976). "Automatic optimum-seeking program for digital simulation." *Simulation*: 27-31.

Smith, D. J. (1990). "The use of mirocomputer-based simulation models in the teaching of operations management." *Int. J. Prod. & Opr. Mang.* **10**: 5-14.

Smith, A. E., Tate, D. M. (1993). Genetic optimisation using a penalty function. Proc. 5th Int. Conf. on GA.

Sofianopoulou, S. (1997). "Application of simulated annealing to a linear model for the formulation of machine cells in group technology." *Int. J. Prod. Res.* **35**(2): 501-511.

Song S. J., Hitomi, K. (1996). "Integrating the production planning and cellular layout for flexible cellular manufacturing" *Production Planning and Control*, **7**: 585-593.

Sonmez, A. I., Baykasoglu, A., Filiz, I. H. (1996). "Computer aided constrained optimisation of cutting conditions in drilling operations on a CNC lathe by using geometric programming." *Mathematical & Computational Applications* **1**(1): 97-104.

Sonmez, A. I., Baykasoglu, A. (1998). "A New Dynamic Programming Formulation of (n*m) flowshop sequencing problems with due dates." *Int. J. of Prod. Res.* **36**(8): 2269-2283.

Sonmez, A. I., Baykasoglu, A., Filiz, I.H., Dereli, T. (1999). "Dynamic optimization of multipass milling operations via geometric programming." *Int. J. of Machine Tools and Manufacture* **39**: 297-320.

Southern, G. (1979). "A factory simulation model for teaching." *Int. J. Mech. Eng. Education* **7**: 183-187.

Spedding, T. A., Lee, W. L., Souza, R., Lee, S. S. G. (1997). "Adaptive simulation of a keyboard assembly cell." *Integ. Manf. Systems* **8**(1): 50-58.

Srinivas, M., Patnaik, L. M. (1994). "Genetic algorithms: A survey." *Computer*: 17-26.

Srinivasan, G., Narendran, T. T., Mahadevan, B. (1990). "An assignment model for the part-families problem in group technology." *Int. J. Prod. Res.* **28**: 145-152.

Stecke, K. E., and Solberg, J. J. (1981). "Loading and control policies for a flexible manufacturing system." *Int. J. of Prod. Res.* **19**(5): 481-490.

Steudel, H. J., Ballakur, A. (1987). "A dynamic programming based heuristic for machine grouping in manufacturing cell formation." *Computers Ind. Engng* **12**(3): 215-222.

Stockon, D. J., Quinn, L. (1993). "Identifying economic order quantities using genetic algorithms." *Int. J. Opr. & Prod. Mang.* **13**: 92-103.

Stockton, D. J., Quinn, L. (1995). "Aggregate production planning using genetic algorithms." *Proc. Instn. Mech. Engrs.* **209**: 201-209.

Stoop, P. P. M., Wiers, V. C. S. (1996). "The complexity of scheduling in practice." *Int. J. of Oper. & Prod. Management* **16**(10): 37-53.

Suer, G. A., Ortega, M. A. (1994). "Machine level based similarity coefficient for forming manufacturing cells." *Computers Ind. Engng* **27**(1-4): 67-70.

Sule, D. R., (1991) "Machine capacity planning in group technology". *Int. J. Prod. Res.*, **29** (9): 1909-1922.

Sun, D., Lin, L., Batta, R. (1995). "Cell formation using tabu search." *Comput. Ind. Engng* **28**(3): 485-494.

Sundaram, R. M. (1978). "An application of goal programming technique in metal cutting." *Int. J. Prod. Research* **16**: 375-382.

Suresh, G., Vinod, V. V., Sahu, S. (1995). "A genetic algorithm for facility layout." *Int. J. Prod. Res.* **33**(12): 3411-3423.

Swain, J. J., Farrington, P. A. (1991). Designing simulation experiments for evaluating manufacturing systems. Proc. of the 1994 Winter Simulation Conf.

Taboun, S. M., Bhole, S. D. (1993). "A simulator for an automated warehousing system." *Comput. Ind. Engng.* **24**(2): 281-290.

Taillard, E. D. (1994). "Parallel taboo search techniques for the job shop scheduling problem." *ORSA Journal on Computing* **6**(2): 108-117.

Tam, K. Y. (1990). "An operation sequence based similarity coefficient for pat families formation." *Journal of Manufacturing Systems* **9**(1): 55-68.

Tang, K. S., Chan, C. Y., Man, K. F., Kwong, S. (1995). Genetic structure for NN topology and weights optimisation. IEE: Gen. Alg. in Eng. Sys.: Innovations and App.

Tautou, L., Pierreval, H. (1995). Using evolutionary algorithms and simulation for the optimisation of manufacturing systems. IEEE Conf. Proc.

Teleb, R., Azadivar, F. (1994). "A methodology for solving multi-objective simulation-optimization problems." *European J. of Oper. Research* **72**: 135-145.

Thesen, A., Travis, L. E. (1991). Introduction to simulation. Proc. of the 1991 Winter Simulation Conf.

Tilsley, R., Lewis, F. A. (1977). "Flexible cell production systems-a realistic approach." *Annals of the CIRP* **25**(1): 269-271.

Tompkins, J. A., White, J. A., Bozer, Y. A., Frazelle, E. H., Tanchoco, J. M. A..

Trevino, J. (1996). Facilities Planning, Jhon Wiley & Sons, Inc.

Ueda, K., Vaario, J., Ohkura, K. (1997). "Modelling of biological manufacturing systems for dynamic reconfiguration." *Annals of the CIRP* **46**: 343-346.

Vaario, J., Ueda, K. (1998). "An emergent modelling method for dynamic scheduling." *J. of Intelligent Manufacturing* **9**: 129-140.

Vaithianathan, R., McRoberts, K. L. (1982). "On scheduling in a GT environment." *J. of Manufacturing Systems* **1**(2): 149-155.

Vakharia, A. S., Wemmerlov, U. (1990). "Designing a cellular manufacturing system: a materials flow approach based on operation sequences." *IEE Transactions* **22**(1): 84-97.

Vakharia, A. S., Kaku, B. K. (1994). "Redesigning a cellular manufacturing system to handle long-term demand changes: a methodology and investigation." *Decision Sciences* **24**(5): 416-433.

Vancza, J., Markus, A. (1991). "Genetic algorithms in process planning." *Computers in Industry* **17**: 181-194.

Venugobal, V., Narendran, T. T. (1992). "A genetic algorithm approach to the machine-component grouping problem with multiple objectives." *Comput. Ind. Engng* **22**(4): 469-480.

Venugobal, V., Narendran, T. T. (1994). "Machine-cell formation through neural network models." *Int. J. Prod. Res.* **32**(9): 2105-2116.

Wang, J., Roze, C. (1997). "Formation of machine cells and part families: a modified p-median model and a comparative study." *Int. J. Prod. Res.* **35**(5): 1259-1286.

Warnecke, H. J. (1993). The Fractal Company, Springer-Verlag.

Watson, E. F., Sadowski, R. P. (1994). Developing and analysing flexible cell systems using simulation. Proc. of the 1994 Winter Simulation Conf.

Wei, J. C., Kern, G. M. (1989). "Commonality analysis: A linear cell clustering algorithm for group technology." *Int. J. Prod. Res.* **27**(12): 2053-2062.

Wei, J. C., Gaither, N. (1990). "A capacity constrained multi-objective cell formation method." *Journal of Manufacturing Systems* **9**(3): 222-232.

Weller, P. R., Summers, R., Thompson, A. C. (1995). Using a genetic algorithm to evolve an optimum input set for a predictive neural network. IEE: Gen. Alg. in Eng. Sys.: Innovations and App.

Wemmerlov, U., Hyer, N. L. (1987). "Research issues in cellular manufacturing." *Int. J. Prod. Res.* **25**(3): 413-431.

Wemmerlov, U., Hyer, N. L. (1989). "Cellular manufacturing in the U.S. industry: a survey of users." *Int. J. Prod. Res.* **27**(9): 1511-1530.

Wemmerlov, U., Johnson, D. J. (1997). "Cellular manufacturing at 46 user plants: implementation experiences and performance improvements." *Int. J. Prod. Res.* **35**(1): 29-49.

Wienholt, W. (1993). A refined genetic algorithm for parameter optimization problems. Proc. 5th Int. Conf. on GA.

Wilhelm, M. R., Ward, T. L. (1987). "Solving quadratic assignment problems by simulated annealing." *IIE Trans.* **3**: 107-119.

Wilson, J. M. (1992). "Approaches to machine load balancing in flexible manufacturing systems." *J. of Opl Res. Soc.* **43**(5): 415-423.

Winston, W. L. (1994). Operations Research: Applications and Algorithms, Int. Thomson Pub.

Wu, H. L., Venugobal, R., Barash, M. M. (1986). "Design of a cellular manufacturing system: A syntactic pattern recognition approach." *Journal of Manufacturing Systems* **5**(2): 81-87.

Wu, S. D., Wysk, R. A. (1989). "An application of discrete-event simulation to on line control and scheduling in flexible manufacturing." *Int. J. Prod. Res.* **27**: 1603-1623.

Xu, H., Wang (Ben), H. (1989). "Part family formation for GT applications based on fuzzy mathematics." *Int. J. Prod. Res.* **27**(9): 1637-1651.

Yamada, T., Nakano, R. (1995). A genetic algorithm with multi-step crossover for job-shop scheduling problems. IEE: Genetic Algorithms in Engineering Systems: Innovations and Applications.

Yang, T., He, Z., Cho, K. K. (1994). "An effective heuristic method for generalized job shop with due dates." *Comput. Ind. Engng.* **26**(4): 647-660.

Yokota, T., Gen, M., Li, Y. X. (1996). "Genetic algorithm for non-linear mixed integer programming problems and its application." *Comput. Ind Engng.* **30**(4): 905-917.

Zhang, C., Wang, H. P. (1993). "Mixed-discrete nonlinear optimization with simulated annealing." *Eng. Opt.* **21**: 277-291.

Zhang, J., Azadivar, F. (1997). Automatic generation of simulation models for optimum design of manufacturing systems. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Zhao, L., Tsujimura, Y., Gen, M. (1997). A hybrid genetic algorithm for the group scheduling problem in a flow line manufacturing cell. 14th Int. Conf. on Prod. Research, Osaka-Japan.

Zhou, M., Askin, R. (1998). "Formation of general GT cells: an operation based approach." *Comput. Ind. Engng* **34**(1): 147-157.

APPENDICES

Appendix I : C/C++ code for Example 2 of Chapter 5

The C/C++ code for the 1st example problem that was explained in Chapter 5 is given below which shows a simple implementation of the proposed multiple objective tabu search algorithm. The random number generator and the *'round'* functions are not given. The overall best solution is not updated in this program, the output simply shows the value of the variables followed by the objective function vector for each best neighbour in each iteration.

```
#include <stdio.h>
#include <malloc.h>

typedef struct _TabuItem
{
   double x1, x2; /* copy of the variables */
} TabuItem;

TabuItem* TabuList[10]; /* circular list of 10 items*/
int TabuIndex;

/* objective vector, this is really an array of three (d+ d-) pairs */

double objective[6];
/* variables themselves, plus some admin*/

double x1, x2, x1_best, x2_best, x1_copy, x2_copy;

/* tells us if the solution is Tabu */

int InTabuList()
{
   int i;
   for(i=0; i<10; i++)
      if(TabuList[i])
         if(TabuList[i]->x1==x1 && TabuList[i]->x2==x2)
            return 1;
```

```
    return 0;
}

/* add the old solution to the Tabu List */

void AddToTabuList()
{
    if(TabuList[TabuIndex])
        free(TabuList[TabuIndex]); /* overwrite old items */


 TabuList[TabuIndex]=(TabuItem*)malloc(sizeof(TabuItem));
    TabuList[TabuIndex]->x1=x1_copy;
  TabuList[TabuIndex]->x2=x2_copy;
    TabuIndex=(TabuIndex+1)%10; /* circular list */
}

/* calculates d+, d- from the actual deviation */

void CalculateDeviations(double dev, double* d_minus, double* d_plus)
{
    if(dev<0.0)
    {
        *d_minus=0.0; *d_plus=0.0-dev;
    }
    else
    {
        *d_minus=dev; *d_plus=0.0;
    }
}

/*Fill in the objective vector,(with the necessary factors)*/

void CalculateObjectiveVector()
{
    double dev;
    dev=0.32 - (0.04*x1 + 0.06*x2);
    CalculateDeviations(dev, &objective[0], &objective[1]);
    dev=0.288 - (0.072*x1 + 0.036*x2);
    CalculateDeviations(dev, &objective[2], &objective[3]);
    dev=0.0 - (x1 - 2.0*x2);
    CalculateDeviations(dev, &objective[4], &objective[5]);
  objective[4]*=2;
}

/* these must not be violated */

int Constraints()
{
    if((3000.0*x1 + 2000.0*x2 <= 16000.0) && (x1 <= 4.0) &&
                              (x2 <= 5.0) && (x1>=0.0) && (x2>=0.0))
        return 1;
    return 0;
}
```

```
/* return 1 if a neighbour can be found, 0 otherwise */
/* update *_best if the neighbour is an improvement */

int EvaluateNeighbour()
{
    int i, step, iter=0;
    double objective_copy[6];

  for(i=0; i<6; i++) /*make a copy of the objective vector*/
    objective_copy[i]=objective[i];
    while(iter++<1000)
    {
        step = round((2.0*random()-1.0)*2.0); /*implement these
         functions yourself !*/

        if(random()<0.5) /* choose a variable at random */
            x1=x1+step;
        else
            x2=x2+step;
        if(Constraints() && !InTabuList())
        {
            CalculateObjectiveVector();
            for(i=0; i<6; i+=2)
            {
            /* if there is an improvement, update the best and
        return */

       if((objective[i]+objective[i+1]) <
                          (objective_copy[i]+objective_copy[i+1]))
            {
               x1_best=x1; x2_best=x2;
               return 1;
            }

      /* if there is an decrease in quality, just return */

       if((objective[i]+objective[i+1]) >
                          (objective_copy[i]+objective_copy[i+1]))
               return 1;
          /* otherwise, keep looking through the objective */
          }
          return 1; /* the objective vectors are identical */
       }
       x1=x1_copy; x2=x2_copy; /*restore old values and loop */
    }
    return 0; /* a neighbour cannot be found */
}

/* the tabu search itself */

void Solve()
{
    int i, iter=0;
    while(iter++ < 500)
    {
```

```
        x1_copy=x1; x2_copy=x2; /*get a copy of the initial
                        values */
        CalculateObjectiveVector();
        printf("%2.2f %2.2f -> ", x1, x2);
        printf("%2.2f %2.2f %2.2f\n", objective[0]+objective[1],
             objective[2]+objective[3],objective[4]+objective[5]);
        for(i=0; i<3; i++) /* evaluate 3 neighbours */
            if(!EvaluateNeighbour(i))
                return;
        AddToTabuList();
        x1=x1_best; x2=x2_best;
    }
}
void main()
{
    int i;
    x1=1.0; x2=1.0; /* initial values for the variables */
    x1_best=x1; x2_best=x2;
    for(i=0; i<10; i++) /* initialise the tabu list */
        TabuList[i]=0;
    TabuIndex=0;
    Solve();
}
```

Appendix II : Additional test problems for Chapter 5

Test Proble A-1

Sundaram (1978) presented the application of goal programming technique in metal cutting applications. He proposed an approach that can be employed for multiple objective optimisation of single pass turning operations. Detailed explanations about model development can be found in his paper. The pre-emptive goal programming model with continuous variables is given as follows;

$$\text{lexmin}\{z_1 = (d_1^+), z_2 = (d_1^-), z_3 = (d_2^+)\}$$
$$s.t.$$
$$3.98624x_1 + 3.47409x_2 + 0.91986x_3 \leq 28.68294$$
$$x_1 + 0.7542x_2 + 0.9005x_3 \leq 9.63939$$
$$x_3 - d_1^+ + d_1^- = 1.38629$$
$$x_1 + x_2 + d_2^+ - d_2^- = 7.07165$$
$$5.65249 < x_1 < 6.52209$$
$$0.40547 < x_2 < 0.73237$$
$$x_1, x_2, x_3, d_1^+, d_1^-, d_2^+, d_2^- \geq 0$$

He gave the following solution: x_1=6.23637, x_2=0.40548, x_3=1.38629, z_1=0, z_2=0.43

The solution obtained from the Tabu search algorithm is as follows:

Parameter Set:	*Stepc*=0.1, *nneigh*=7, *m*=10, *iter*=8000, *t*=100
Computation time:	5.3 sec.
Solution:	x_1=5.84832, x_2=0.631812, x_3=1.38629, z_1=0, z_2=0 (optimum)

Tabu search successfully found the optimum solution (i.e. all deviations are equal to zero) for this test problem.

Test problem A-2

Schniederjans and Kwak (1982) proposed an alternative method for solving goal programming problems. Their procedure is based on Baumol's simplex method for solving linear programming problems with minor modifications. In their paper they solved the following test problem optimally.

$$\text{lexmin}\{z_1 = (d_1^-), z_2 = (d_4^+), z_3 = (5d_2^-), z_4 = (3d_3^-), z_5 = (d_1^+)\}$$
$$s.t.$$
$$x_1 + x_2 + d_1^- - d_1^+ = 80$$
$$x_1 + d_2^- = 70$$
$$x_2 + d_3^- = 45$$
$$x_1 + x_2 + d_4^- - d_4^+ = 90$$
$$x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+ \geq 0$$

The optimum solution is: x_1=70, x_2=20, z_1=0, z_2=0, z_3=0, z_4=75, z_5=10

The solution obtained from the Tabu search algorithm is as follows:

Parameter Set:	*Stepi*=2, *nneigh*=10, *m*=10, *iter*=8000, *t*=100
Computation time:	4.2 sec.
Solution:	x_1=70, x_2=20, z_1=0, z_2=0, z_3=0, z_4=75, z_5=10 (optimum)

Test problem A-3

The following test problem is taken from Ignizo (1982)'s book. It represents an integer goal programming formulation of a product mix problem. The problem was solved optimally by the simplex method.

$$\text{lexmin}\{z_1 = (d_1^- + d_2^- + d_3^+), z_2 = (d_4^+), z_3 = (d_5^-)\}$$

$$s.t.$$

$$x_1 + x_2 + d_1^- - d_1^+ = 30$$

$$x_3 + x_4 + d_2^- - d_2^+ = 30$$

$$3x_1 + 2x_3 + d_3^- - d_3^+ = 120$$

$$3x_2 + 2x_4 + d_4^- - d_4^+ = 20$$

$$10x_1 + 9x_2 + 8x_3 + 7x_4 + d_5^- - d_5^+ = 800$$

$$x_i \geq 0 \quad i = 1,\ldots,4$$

$$d_i^-, d_i^+ \geq 0 \quad i = 1,\ldots,5$$

The optimum solution is: x_1=20, x_2=10, x_3=30, x_4=0, z_1=0, z_2=10, z_3=270

The solution obtained from the Tabu search algorithm is as follows:

Parameter Set:	*Stepi*=2, *nneigh*=10, *m*=10, *iter*=8000, *t*=100
Computation time:	4.3 sec.
Solution:	x_1=20, x_2=10, x_3=30, x_4=0, z_1=0, z_2=10, z_3=270 (optimum)

Test problem A-4

The following integer linear pre-emptive goal programming example is adopted from LINDO optimisation software's manuals. It represents a staff-scheduling model. LINDO software uses a simplex-based method. LINDO solved the problem around 4 seconds optimally.

$$\text{lexmin}\{z_1 = (d_1^+), z_2 = (d_2^-)\}$$

$s.t.$

$$9x_1 + 9x_2 + 9x_3 + 9x_4 + 9x_5 + 9x_6 + 9x_7 + d_1^- - d_1^+ = 0$$

$$x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + d_2^- - d_2^+ = 7$$

$$x_1 + x_2 + x_3 + x_4 + x_5 - x_8 \geq 3$$

$$x_1 + x_3 + x_4 + x_5 + x_6 - x_9 \geq 3$$

$$x_1 + x_4 + x_5 + x_6 + x_7 - x_{10} \geq 8$$

$$x_1 + x_2 + x_5 + x_6 + x_7 - x_{11} \geq 8$$

$$x_1 + x_2 + x_3 + x_6 + x_7 - x_{12} \geq 8$$

$$x_2 + x_3 + x_4 + x_6 + x_7 - x_{13} \geq 3$$

$$x_2 + x_3 + x_4 + x_5 + x_7 - x_{14} \geq 3$$

$$x_8 \leq 1$$

$$x_9 \leq 1$$

$$x_{10} \leq 1$$

$$x_{11} \leq 1$$

$$x_{12} \leq 1$$

$$x_{13} \leq 1$$

$$x_{14} \leq 1$$

$$x_i \geq 0 \quad i = 1,....,7$$

$$x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14} = 0 \text{ or } 1$$

$$d_1^-, d_1^+, d_2^-, d_2^+ \geq 0$$

The optimum solution is: x_1= 4, x_2=0, x_3=0, x_4=0, x_5=0, x_6=0, x_7=4, x_8=1, x_9=1, x_{10}=0, x_{11}=0, x_{12}=0, x_{13}=1, x_{14}=1, z_1=72, z_2=3

The solution obtained from the Tabu search algorithm is as follows:

Parameter Set:	*Stepi*=2, *nneigh*=10, *m*=10, *iter*=8000, *t*=100
Computation time:	11.2 sec.
Solution:	x_1= 4, x_2=0, x_3=0, x_4=0, x_5=0, x_6=0, x_7=4, x_8=1, x_9=1, x_{10}=0, x_{11}=0, x_{12}=0, x_{13}=1, x_{14}=1 , z_1=72, z_2=3 (optimum)

Test problem A-5

Schniederjans and Santhanam (1989) proposed a zero-one linear pre-emptive goal programming formulation for the journal selection and cancellation problem. They used Boumol's simplex method in their solution. Detailed explanation about model formation can be found in their paper. The mathematical model is given as follows.

$$\text{lexmin}\{z_1=(d_1^-), z_2=(d_4^-), z_3=(d_2^+ + d_3^+), z_4=(d_5^+), z_5=(d_6^+)\}$$

s.t.

$$300x_1 + 220x_2 + 400x_3 + 700x_4 + 350x_5 + 260x_6 + 270x_7 + 360x_8 + 250x_9 + 210x_{10} + 260x_{11} + 320x_{12} + 200x_{13} + 520x_{14} + 200x_{15} + d_1^- - d_1^+ = 2000$$

$$200x_1 + 50x_2 + 400x_3 + 60x_4 + 70x_5 + 160x_6 + 351x_7 + 130x_8 + 85x_9 + 70x_{10} + 215x_{11} + 45x_{12} + 66x_{13} + 130x_{14} + 312x_{15} + d_2^- - d_2^+ = 0$$

$$110x_1 + 120x_2 + 200x_3 + 80x_4 + 100x_5 + 210x_6 + 152x_7 + 111x_8 + 95x_9 + 65x_{10} + 98x_{11} + 35x_{12} + 43x_{13} + 120x_{14} + 110x_{15} + d_3^- - d_3^+ = 0$$

$$5x_1 + 5x_2 + 2x_3 + 4x_4 + 3x_5 + 5x_6 + 2x_7 + 1x_8 + 6x_9 + 5x_{10} + 3x_{11} + 4x_{12} + 5x_{13} + 4x_{14} + 1x_{15} + d_4^- - d_4^+ = 9999$$

$$70x_1 + 70x_2 + 90x_3 + 90x_4 + 60x_5 + 90x_6 + 105x_7 + 85x_8 + 70x_9 + 40x_{10} + 65x_{11} + 45x_{12} + 50x_{13} + 70x_{14} + 90x_{15} + d_5^- - d_5^+ = 0$$

$$2x_1 + 1x_2 + 2x_3 + 2x_4 + 2x_5 + 3x_6 + 2x_7 + 2x_8 + 2x_9 + 3x_{10} + 2x_{11} + 1x_{12} + 2x_{13} + 3x_{14} + 4x_{15} + d_6^- - d_6^+ = 0$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15} = 0 \text{ or } 1$$

$$d_1^-, d_1^+, d_2^-, d_2^+, d_3^-, d_3^+, d_4^-, d_4^+, d_5^-, d_5^+, d_6^-, d_6^+ \geq 0$$

Their solution is: x_1=0, x_2=0, x_3=0, x_4=1, x_5=0, x_6=0, x_7=0, x_8=0, x_9=1, x_{10}=1, x_{11}=0, x_{12}=1, x_{13}=0, x_{14}=1, x_{15}=0, z_1=0, z_2=9976, z_3=785, z_4=275, z_5=11

The solution obtained from the Tabu search algorithm is as follows:

Parameter Set:	*nneigh*=10, *m*=10, *iter*=8000, *t*=100
Computation time:	13.5 sec.
Solution:	x_1=1, x_2=1, x_3=1, x_4=1, x_5=1, x_6=1, x_7=1, x_8=1, x_9=1, x_{10}=1, x_{11}=1, x_{12}=1, x_{13}=1, x_{14}=1, x_{15}=1, z_1=0, z_2=9944, z_3=3993, z_4=1090, z_5=33

The solution obtained from the Tabu search algorithm satisfied the first goal completely and the satisfaction level for the second most important goal is better than the given solution. Satisfaction levels for other goals are not as good as the given solution. However, the objective of preemptive goal programming is to satisfy the more important goals first. Therefore the solution obtained from the Tabu search algorithm is the better solution.

Appendix III : Resource Elements Concept

The way in which the capabilities of resources are defined plays an important role in designing, planning, controlling and the efficiency of the production. The classical way of defining manufacturing systems and their capabilities does not provide sufficient level of detail in describing shared and unique boundaries between production resources. The main idea of the Resource Elements (RE) is to define the shared and unique capabilities of production resources. The operation level is very detailed and the full resource level (i.e. machine level) is too general. REs allow the inherent flexibility available in a machine shop to be revealed and therefore can be better utilised.

REs have been proposed by Gindy *et. al.* (1996) in order to define machine tool capabilities and part processing requirements with a common language. This enables better modelling of various manufacturing design and operating functions. Moreover, using REs can result in seamless integration of several manufacturing functions such as process planning and scheduling (Carvalho and Gindy, 1995), process planning and loading (Baykasoglu, Saad and Gindy, 1998).

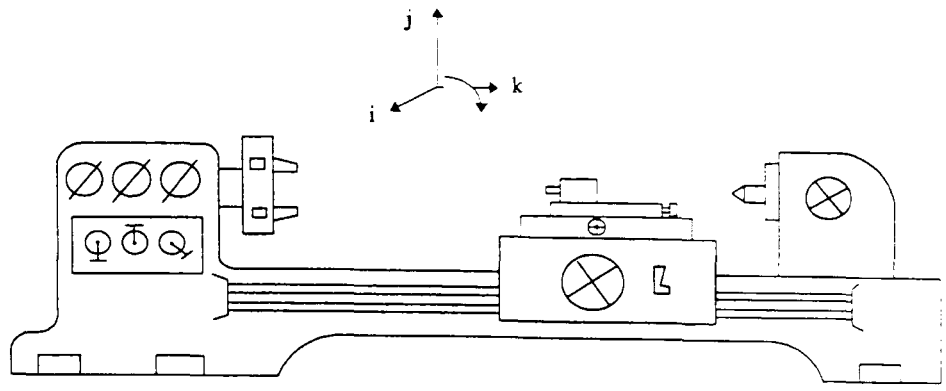
Machine Resources and Form Generating Schemas

Machines are the main resources in any manufacturing facility. The capability boundaries of each machine are very well known and the operations that they can perform are also very well known. Nevertheless, some operations can be performed on more than one machine, and some machines can perform many different operations (e.g. a machining centre combines capabilities of a lathe and a milling

machine). Therefore, in planning applications, representing component requirements in terms of whole machines is inappropriate as this does not show the availability of possible multiple routings.

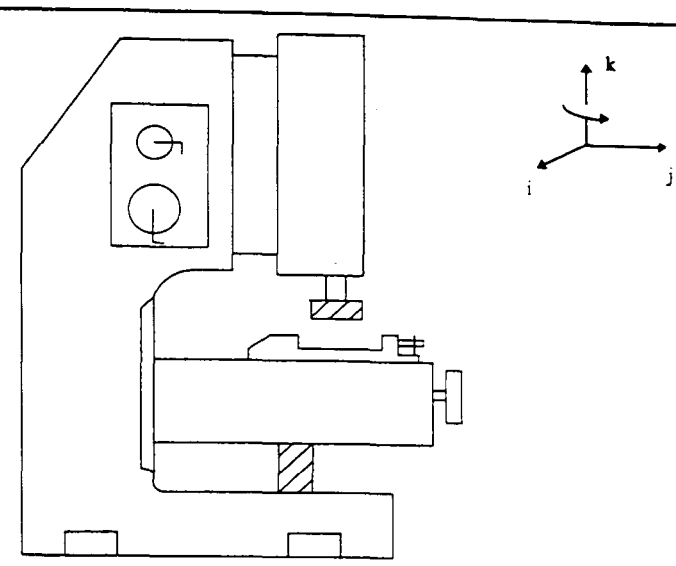
There is a close relationship between the capabilities of machines and the operations they can perform. Machining operations give shape to a workpiece by removing material from predefined locations on that workpiece. A potential machining operation is represented as a Form Generating Schema (FGS). A machining operation (drilling, turning etc.) is defined only when the machine where this operation is going to be performed is selected.

FGSs are generic machine-independent capability patterns and they are used for representing processing requirements of parts and capabilities of machine tools. A FGS is a technologically meaningful combination of a cutting tool of a specific geometry, a set of relative motions between part and the cutting tool, and the typical levels of technological output that can be associated with using that combination of tool and relative motion (Gindy, *et. al.* 1996). Several examples of FGS's that can be obtained from a milling machine and a classical lathe are shown in Figures III.1 and III.2.



Cutting Tool	Motion Set	FGS
*External Roughing Tools	Rk Tk ti	External Rough Turning
*External Finishing Tools	Rk Tk Ti ti	Ext. Prof. Turn. (Tapering)
	Rk Ti tk	Facing
	Rk Tk ti	Ext. Finish Turning
*Internal Roughing Tools	Rk Tk ti	Internal Rough Turning
*Internal Finishing Tools	Rk Tk Ti ti	Int.. Prof. Turn. (Tapering)
	Rk Tk ti	Int. Finish Turning
*External Threading Tools	Rk Tk ti	Ext. Threading
*Internal Threading Tools	Rk Tk ti	Int. Threading
*Drill	Rk Tk	Drilling
*Core Drills	Rk Tk	Core Drilling
*Step Drills	Rk Tk	Step Drilling
*Counterbore	Rk Tk	Counter boring
*Center Drills	Rk Tk	Center Drilling
*Gun Drills	Rk Tk	Gun Drilling
*Countersink	Rk Tk	Countersinking
*Reamers	Rk Tk	Reaming
*Grooving Tools	Rk Ti	Grooving or Cutting off
	Rk Ti tk	Recess Grooving
	Rk Tk	Face Grooving
*Internal Grooving Tools	Rk Ti	Internal Grooving
	Rk Ti tk	Internal Recess Grooving
*Knurling Tools	Rk Tk ti	Knurling
*Ext. Form Cutting Tools	Rk Ti	External Forming
*Int. Form Cutting Tools	Rk Ti	Internal Forming
*Chamfer Tool	Rk Tk ti	Chamfering

Figure III.1 FGSs in a classical lathe



Cutting Tool	Motion Set	FGS
*Drill	Rk Tk	Drilling
*Core Drills	Rk Tk	Core Drilling
*Step Drills	Rk Tk	Step Drilling
*Counterbors	Rk Tk	Counter Boring
*Center Drills	Rk Tk	Center Drilling
*Gun Drills	Rk Tk	Gun Drilling
*Countersink	Rk Tk	Countersinking
*Reamers	Rk Tk	Reaming
*Boring Bars	Rk Tk ti	Boring
*Slot Drills	Rk Tk	Slot Drilling-1
	Rk Ti tk	Slot Drilling-2
	Rk Ti Tj tk	Slot Drilling-3
	Rk Ti tj	Slot Drilling-4
*End Mills	Rk Ti tk	End Milling-1
	Rk Ti tj	End Milling-2
	Rk Ti Tj tk	End Milling-3
*Face Mills	Rk Ti tk	Face Milling

Figure III.2 FGSs in a milling machine

In Figures III.1 and III.2: R, T designates formative motions (R for rotations and T for translations), r, t designates positioning motions (r for rotations and t for translations) and i, j, k are motion axes.

Once a FGS is allocated to a machine tool, it becomes a machining operation. The way in which a FGS is executed is dependent on the machine tool that the FGS is

attached to (Gindy *et. al.*, 1996). Although the relative motion set and the tool used are the same, the way in which the operation is executed may differ from machine to machine (e.g. execution of drilling operation on Lathe and on a Drilling Machine), and the level of technological output may also differ.

Resource Elements (RE)

A machining facility would be better defined as a collection of FGSs that are REs. The group of resources contained in a manufacturing facility is described by using a set of REs. Each RE represents a collection of FGSs and is such that the exclusive and the shared capability boundaries between the available resources contained in a manufacturing facility are uniquely identified (Gindy *et. al.*, 1996) (see Figure III.3).

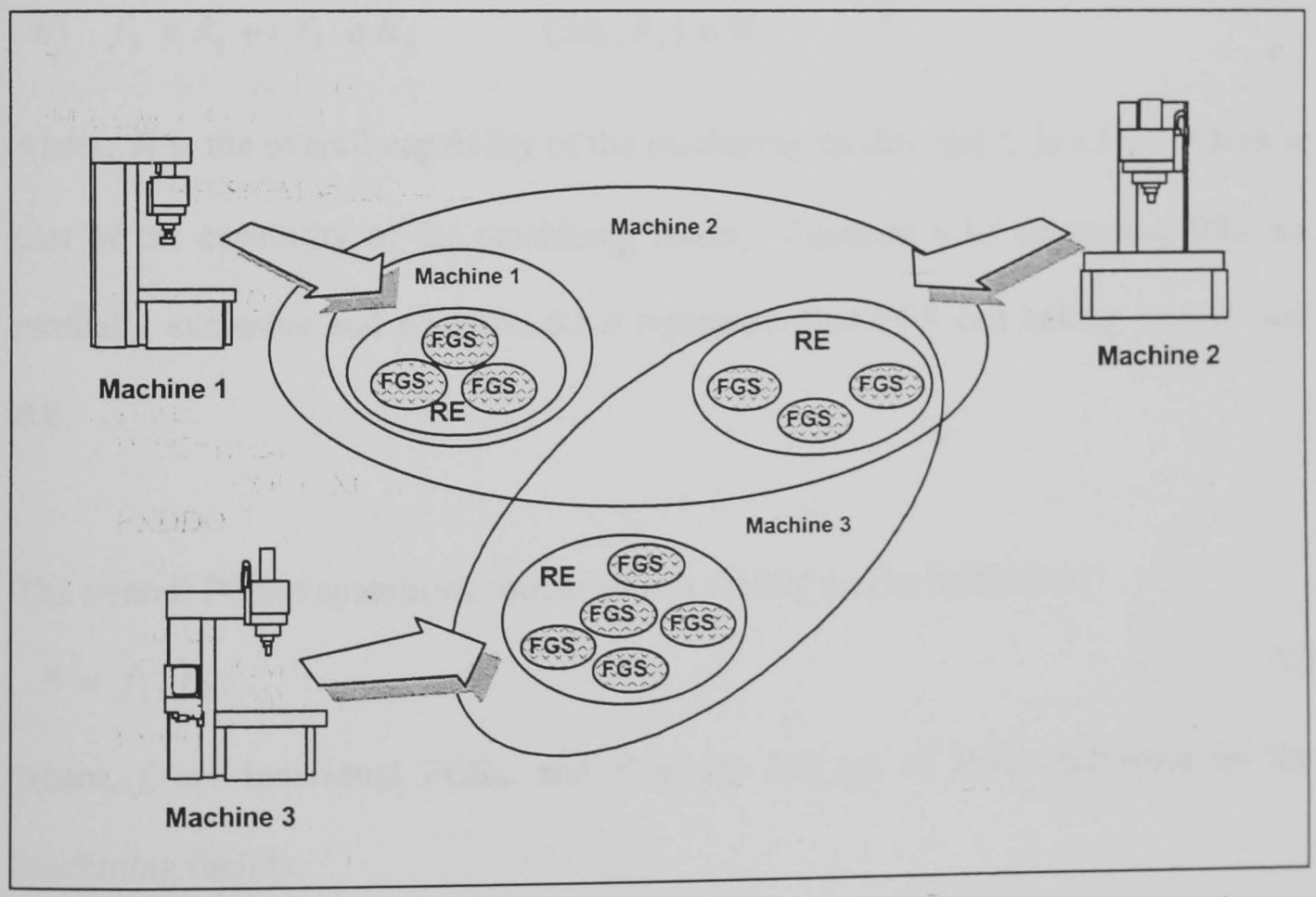


Figure III.3 RE based representation of a machining facility with three machines

Capability Based Representation of a Machining Facility

Considering that, for a particular machining facility, there is a group of FGSs that do not appear separately, i.e. a resource containing a certain FGS, belonging to that group, also holds all the other FGSs of the same group. In this case that group of FGSs can be considered as a basic unit. This group of FGSs is considered to be a RE. This is the basic principle of Resource Elements. A machining facility can be either defined as a list of FGSs or, more effectively, as a list of REs. The number and the capability of each one of the REs is specific for each machining facility.

RE R_i in $\Re$ is formally defined as (Gindy *et. al,* 1996):

$$\begin{aligned} &i) \quad R_i \cap R_j \equiv 0 && (\forall R_i, R_j) \in \Re \\ &ii) \quad f_k \in R_i \leftrightarrow f_k \notin R_j && (\exists R_i, R_j) \in \Re \end{aligned} \qquad i \neq j \qquad 6.1$$

where, $\Re$ is the overall capability of the machining facility and f_k is a FGS, which is part of the capability of the machining facility. Equation 6.1.*i* shows that REs are mutually exclusive and equation 6.1.*ii* represents that FGS can belong to only one RE.

The overall FGSs capabilities contained in a facility can be defined as:

$$F = f_1, f_2, \ldots\ldots\ldots\ldots, f_n \qquad 6.2$$

where, f_i are individual FGSs, and F is the full set of FGS performed by the machining facility.

A resource is represented by the set of FGSs. The capability of a resource M_i in a facility M, can be shown by a vector:

$$M_k = m_{1k}, m_{2k}, \ldots\ldots m_{nk} \qquad M_k \in M \tag{6.3}$$

where, m_{ik} is equal to 1 if FGS f_i belongs to resource k, 0 otherwise.

The REs in a machining facility are defined by an iterative procedure (Gindy *et. al.*, 1996). For each two FGSs f_p and f_q in F:

$$IF \quad \forall M_k \in M : m_{pk} = m_{qk}, THEN \quad cluster \quad f_p \& f_q \ together \tag{6.4}$$

Each RE R_j is represented by a vector:

$$R_j = r_{1j}, r_{2j}, \ldots\ldots, r_{nj} \quad R_j \in R \tag{6.5}$$

where, r_{ij} is 1 if f_i is a part of RE R_j, 0 otherwise.

The uniqueness of the REs is defined by;

$$\sum_k r_{ik} = 1 \qquad \forall i = 1,2 \ldots n \tag{6.6}$$

Equation 6.6 guarantees that each FGS belongs to only one RE and that there is no overlapping between RE. The algorithm for dividing a machining facility into a set of REs is given as follows (Gindy *et. al.*, 1996).

Step-1: Define the full set of FGSs in F.

Step-2: Define the capability of each resource in the facility using FGSs.

Step-3: Select FGS f_i.

Step-4: Select f_j $(j \neq i)$.

Step-5: If $m_{ik} = m_{jk}$ for each resource in the facility $M_k(k=1,2,\ldots m)$ then cluster together f_i &

Step-6: Repeat steps 3,4 and 5 for all $i, j=1,2,\ldots n$.

Step-7: Define each cluster of FGS as a RE.

Step-8: Using the clusters of FGSs, represent each resource in the facility as a set of REs.

The REs defined for a resource are dependent on the capabilities of the other resources in the machining facility. The distribution of REs for the same resource can change when other resources are either added into or removed from the machining facility. Once the above methodology has been applied, the machining facility is no longer seen as a group of resources. Instead, the FGSs of a machining facility can now be described by the list of REs it contains and the number of repetitions of REs give an indication of the flexibility of the manufacturing facility. This concept simplifies and enhances manufacturing tasks such as production planning and control, process planning, shop floor design, scheduling etc. and also allows for accurate comparison between the components technological needs and the manufacturing capabilities available in the manufacturing facility (Gindy *et. al.*, 1996).

The main characteristic features of REs can be summarised and listed as follows:

- REs are mutually exclusive, in other words there is no overlap between REs they are unique (i.e. set of FGS in each RE are totally different and each FGS can belong to only one RE). Some other important characteristics of REs are as follows;
- A resource that provides a RE is capable of performing all tasks within that RE.
- A resource may provide many REs or only one RE.
- A component requiring a RE has the change to access all the resources that provide that RE.
- REs are unique planning and scheduling entities

Example

Consider a manufacturing cell with seven machines and their FGSs are shown in Table III.1.

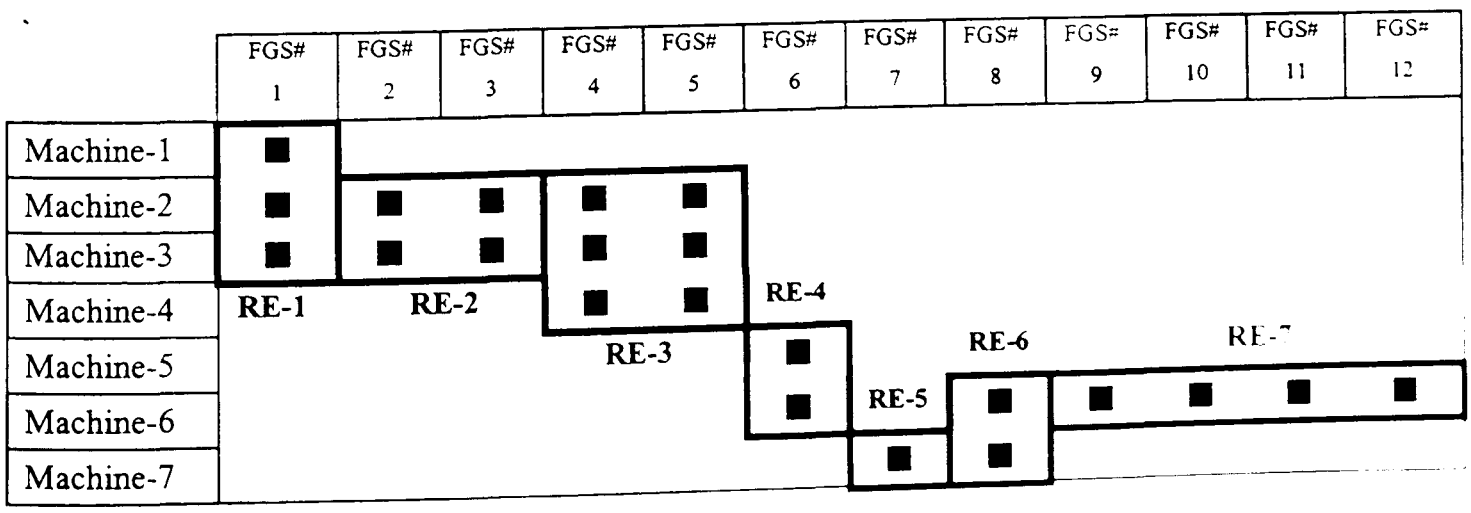
Table III.1 The capability matrix M_k for the example manufacturing cell

	FGS# 1	FGS# 2	FGS# 3	FGS# 4	FGS# 5	FGS#6	FGS# 7	FGS# 8	FGS# 9	FGS# 10	FGS# 11	FGS# 12
Machine-1	■											
Machine-2	■	■	■	■	■							
Machine-3	■	■	■	■	■							
Machine-4				■	■							
Machine-5						■						
Machine-6						■		■	■	■	■	■
Machine-7							■	■				

FGS#1: Drilling, FGS#2: Turning, FGS#3: Boring, FGS#4: Grooving .....

By applying the clustering procedure (Gindy *et. al.*, 1996) that is defined above, the REs based representation of the manufacturing cell can be obtained. The procedure starts with the clustering of each pair of FGSs in the facility F complying with the conditions defined in the algorithm and gradually increasing the size of the clusters until all possible combinations of FGSs are considered. The resulting REs for the example machining cell are shown in Table III.2 the schematic representation of the cell is given in Figure III.4.

Table III.2 Clustering FGSs to form REs for the example manufacturing cell



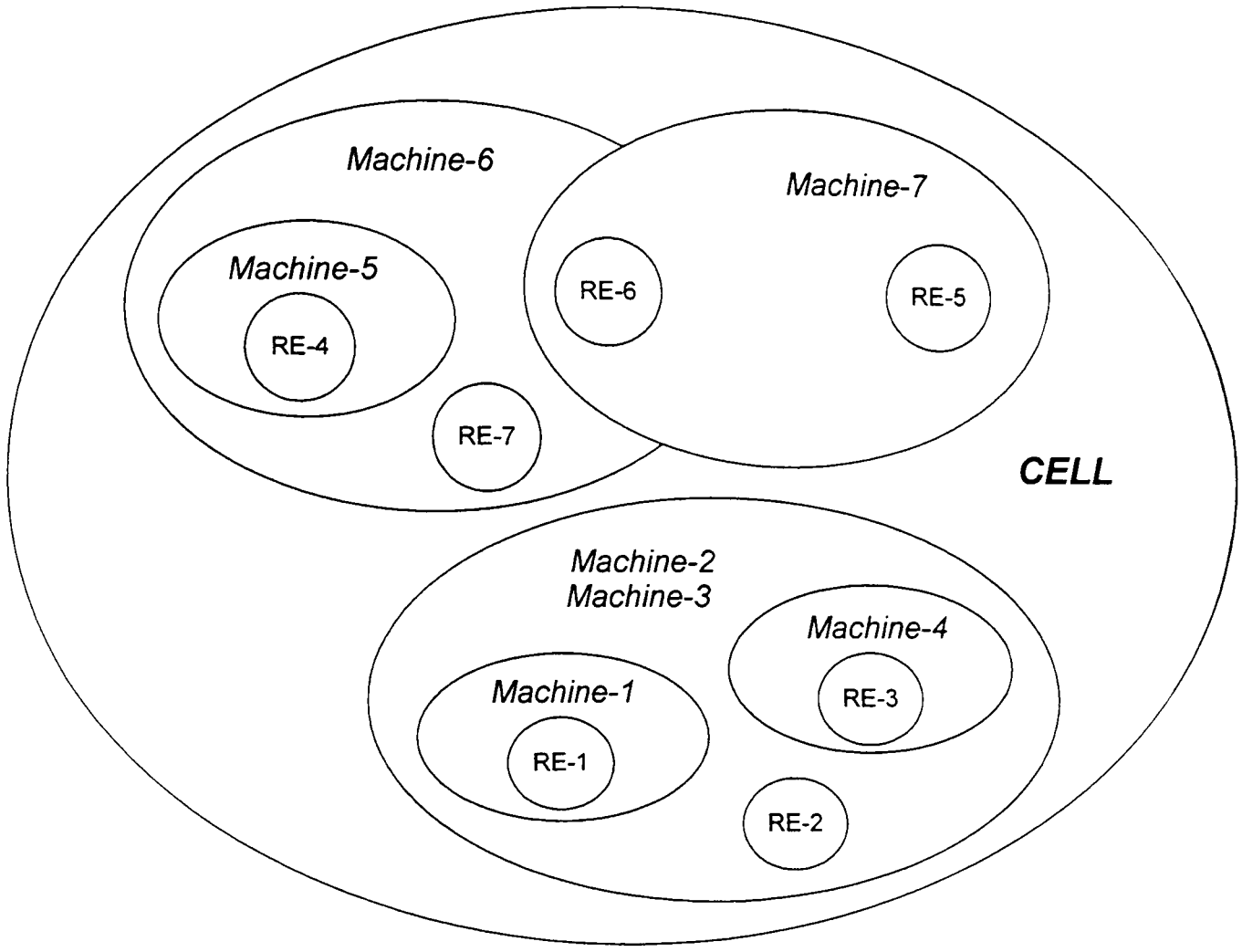


Figure III.4 REs based representation of the manufacturing cell

Appendix IV : FORTRAN-90 code for the cell configuration module

The FORTRAN-90 code for the tabu search based multiple objective capability based concurrent cell configuration model (MOCACEF 1.0) is given below. An example application with real manufacturing data is also given in Appendix V with the corresponding data files.

```
PROGRAM MOCACEF 1.0

INTEGER Z(100),X(100,10),Y(100,10),XOPT(100,10),YOPT(100,10)
INTEGER PMIN,PMAX,MMAX,ZOPT(10),Z1(10),X1(100,10),Y1(100,10)
INTEGER P(100,50),M(100,50),NRE,P_QAN(100),TOTCRE(20,50)
INTEGER I,J,K,RE,ISEED,NPART,NMACH,GMAX,MMIN,Y3(100,10)
INTEGER Z2(10),X2(100,10),Y2(100,10),Z3(10),X3(100,10)
INTEGER COPT_X(100,10),COPT_Y(100,10),COPT_Z(10),DUMMY
INTEGER TALIST_P(40,100),TALIST_M(40,100),N_ITER,TOTCRE3(20,50)
INTEGER TABUSIZE,COUNTER,NRESPONS,GXYZTOP,GXYITOP,Y4(100,10)
INTEGER TOTCRE1(20,50),TOTCRE2(20,50),TOTCRE4(20,50)
INTEGER C_TOTCRE(20,50),O_TOTCRE(20,50),Z4(100),X4(100,10)
REAL M_CAP(50),P_MT(100,100),DSM(100,100),CBESTV(10)
REAL BEST_V(10),RES_V(10),RES_V1(10),RES_V2(10),RES_V3(10)
REAL GOAL1,GOAL2,GOAL3,GOAL4,OBJ1,OBJ2_3,OBJ4,OBJ5
REAL CL(20),CAP_FAC(20),CAP_FAC1(20,50),RES_V4(10)
REAL CL1(20),CELLCAP1(20),RECAP1(20,50)
REAL CL2(20),CELLCAP2(20),RECAP2(20,50)
REAL CL4(20),CELLCAP4(20),RECAP4(20,50)
REAL CL3(20),CELLCAP3(20),RECAP3(20,50)
REAL C_CL(20),C_CELLCA(20),C_RECAP(20,50)
REAL O_CL(20),O_CELLCA(20),O_RECAP(20,50)

OPEN(UNIT=7,STATUS='UNKNOWN',FILE='C:\RECON_OUT\TS_CONVG.TXT')
OPEN(UNIT=1,STATUS='OLD',FILE='C:\RECON_IN\JOB_RE.TXT')
OPEN(UNIT=2,STATUS='OLD',FILE='C:\RECON_IN\MACH_RE.TXT')
OPEN(UNIT=3,STATUS='OLD',FILE='C:\RECON_IN\MACH_CAP.TXT')
OPEN(UNIT=4,STATUS='OLD',FILE='C:\RECON_IN\PRO_TIME.TXT')

C
C    DATA INPUT FROM DATA FILES
C

READ(1,*) NPART,NRE
DO I=1,NPART
        READ(1,*) (P(I,J),J=1,NRE)
ENDDO

READ(2,*) NMACH,NRE
DO I=1,NMACH
        READ(2,*) (M(I,J),J=1,NRE)
ENDDO

DO I=1,NMACH
        READ(3,*) M_CAP(I)
ENDDO
```

```
DO I=1,NPART
        READ(4,*) (P_MT(I,J),J=1,NRE)
ENDDO

DO I=1,NPART
        READ(4,*) P_QAN(I)
ENDDO

CLOSE(UNIT=1)
CLOSE(UNIT=2)
CLOSE(UNIT=3)
CLOSE(UNIT=4)

READ*, MMIN,MMAX,PMIN,PMAX,GMAX
READ*, TABUSIZE,N_ITER

NRESPONS=4
C   READ*, GOAL1,GOAL2,GOAL3

GOAL1=0
GOAL2=0
GOAL3=0
GOAL4=REAL(NRE*GMAX)

ISEED=9999

CALL DISSIMIL(P,NPART,NRE,DSM)
CALL INRANSOL(ISEED,GMAX,NPART,NMACH,P,M,DSM,NRE,MMIN,MMAX,PMIN
+ ,PMAX,M_CAP,P_MT,P_QAN,X,Y,Z)

C  CALL INFESS(GMAX,X,Y,Z)

C
C  INITIALISATION
C

DO I=1,NPART
        DO K=1,GMAX
                COPT_X(I,K)=X(I,K)
        ENDDO
ENDDO

DO J=1,NMACH
        DO K=1,GMAX
                COPT_Y(J,K)=Y(J,K)
        ENDDO
ENDDO

DO K=1,GMAX
        COPT_Z(K)=Z(K)
ENDDO

CALL OBJFUNC(X,Y,Z,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
+ MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
+ OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

C
C    CALCULATION OF DEVIATION VECTORS
C
```

```
RES_V(1)=ABS(GOAL1-OBJ1)
RES_V(2)=OBJ2_3
RES_V(3)=ABS(GOAL3-OBJ4)
RES_V(4)=GOAL4-OBJ5

C
C    INITIALISATION
C

DO I=1,NRESPONS
        BEST_V(I)=RES_V(I)
ENDDO

DO I=1,NPART
        DO K=1,GMAX
                TALIST_P( 1,((GMAX-1)*(I-1))+((I-1)+K) )=COPT_X(I,K)
        ENDDO
ENDDO

DO J=1,NMACH
        DO K=1,GMAX
                TALIST_M( 1,((GMAX-1)*(J-1))+((J-1)+K) )=COPT_Y(J,K)
        ENDDO
ENDDO

C
C    STARTING TABU SEARCH
C

COUNTER=1;
 DO
        COUNTER=COUNTER+1

        CALL MOVEMENT(ISEED,GMAX,NPART,NMACH,P,M,DSM,NRE,MMIN,MMAX
        + ,PMIN,PMAX,M_CAP,P_MT,P_QAN,TABUSIZE,TALIST_P,TALIST_M
        + ,COPT_X,COPT_Y,COPT_Z,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)

        CALL OBJFUNC(X1,Y1,Z1,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
        + MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
        + OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

        DO K=1,GMAX
                DO RE=1,NRE
                        RECAP1(K,RE)=CAP_FAC1(K,RE)
                        TOTCRE1(K,RE)=TOTCRE(K,RE)
                ENDDO
                CL1(K)=CL(K)
                CELLCAP1(K)=CAP_FAC(K)
        ENDDO

C    CALCULATION OF DEVIATIONS

        RES_V1(1)=ABS(GOAL1-OBJ1)
        RES_V1(2)=OBJ2_3
        RES_V1(3)=ABS(GOAL3-OBJ4)
        RES_V1(4)=GOAL4-OBJ5

        CALL OBJFUNC(X2,Y2,Z2,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
        + MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
        + OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)
        DO K=1,GMAX
```

```
            DO RE=1,NRE
                  RECAP2(K,RE)=CAP_FAC1(K,RE)
                  TOTCRE2(K,RE)=TOTCRE(K,RE)
            ENDDO
            CL2(K)=CL(K)
            CELLCAP2(K)=CAP_FAC(K)
ENDDO

RES_V2(1)=ABS(GOAL1-OBJ1)
RES_V2(2)=OBJ2_3
RES_V2(3)=ABS(GOAL3-OBJ4)
 RES_V2(4)=GOAL4-OBJ5

CALL OBJFUNC(X3,Y3,Z3,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
+ MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
 + OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

DO K=1,GMAX
            DO RE=1,NRE
                  RECAP3(K,RE)=CAP_FAC1(K,RE)
                  TOTCRE3(K,RE)=TOTCRE(K,RE)
            ENDDO
            CL3(K)=CL(K)
            CELLCAP3(K)=CAP_FAC(K)
 ENDDO

RES_V3(1)=ABS(GOAL1-OBJ1)
RES_V3(2)=OBJ2_3
RES_V3(3)=ABS(GOAL3-OBJ4)
 RES_V3(4)=GOAL4-OBJ5

            CALL OBJFUNC(X4,Y4,Z4,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
            + MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
            + OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

            DO K=1,GMAX
                  DO RE=1,NRE
                        RECAP4(K,RE)=CAP_FAC1(K,RE)
                        TOTCRE4(K,RE)=TOTCRE(K,RE)
                  ENDDO
                  CL4(K)=CL(K)
                  CELLCAP4(K)=CAP_FAC(K)
            ENDDO

            RES_V4(1)=ABS(GOAL1-OBJ1)
            RES_V4(2)=OBJ2_3
            RES_V4(3)=ABS(GOAL3-OBJ4)
             RES_V4(4)=GOAL4-OBJ5

C
C     SELECTION OF CURRENT BEST SOLUTION VECTOR
C

            DO I=1,NRESPONS
                  CBESTV(I)=RES_V1(I)
            ENDDO

            DO I=1,NPART
                  DO K=1,GMAX
                        COPT_X(I,K)=X1(I,K)
                  ENDDO
```

```
ENDDO

DO J=1,NMACH
        DO K=1,GMAX
                COPT_Y(J,K)=Y1(J,K)
        ENDDO
ENDDO

DO K=1,GMAX
        COPT_Z(K)=Z1(K)
        C_CELLCA(K)=CELLCAP1(K)
        C_CL(K)=CL1(K)
ENDDO

 DO K=1,GMAX
        DO RE=1,NRE
                C_RECAP(K,RE)=RECAP1(K,RE)
                C_TOTCRE(K,RE)=TOTCRE1(K,RE)
         ENDDO
 ENDDO

 DUMMY=0
 IF(RES_V2(1).LT.CBESTV(1)) THEN
         DUMMY=DUMMY+1
 ELSEIF(RES_V2(1).EQ.CBESTV(1).AND.RES_V2(2).LT.CBESTV(2)) THEN
         DUMMY=DUMMY+1
 ELSEIF(RES_V2(1).EQ.CBESTV(1).AND.RES_V2(2).EQ.CBESTV(2)
 + .AND.RES_V2(3).LT.CBESTV(3)) THEN
         DUMMY=DUMMY+1
 ELSEIF(RES_V2(1).EQ.CBESTV(1).AND.RES_V2(2).EQ.CBESTV(2)
 + .AND.RES_V2(3).EQ.CBESTV(3).AND.RES_V2(4).LE.CBESTV(4)) THEN
         DUMMY=DUMMY+1
ENDIF

IF(DUMMY.GE.1) THEN
        DO I=1,NRESPONS
                CBESTV(I)=RES_V2(I)
         ENDDO

        DO I=1,NPART
                DO K=1,GMAX
                        COPT_X(I,K)=X2(I,K)
                ENDDO
        ENDDO

        DO J=1,NMACH
                DO K=1,GMAX
                        COPT_Y(J,K)=Y2(J,K)
                ENDDO
        ENDDO

        DO K=1,GMAX
                COPT_Z(K)=Z2(K)
                C_CELLCA(K)=CELLCAP2(K)
                C_CL(K)=CL2(K)
        ENDDO

        DO K=1,GMAX
                DO RE=1,NRE
                        C_RECAP(K,RE)=RECAP2(K,RE)
                        C_TOTCRE(K,RE)=TOTCRE2(K,RE)
```

```
                ENDDO
        ENDDO
ENDIF

DUMMY=0
IF(RES_V3(1).LT.CBESTV(1)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V3(1).EQ.CBESTV(1).AND.RES_V3(2).LT.CBESTV(2)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V3(1).EQ.CBESTV(1).AND.RES_V3(2).EQ.CBESTV(2)
+.AND.RES_V3(3).LT.CBESTV(3)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V3(1).EQ.CBESTV(1).AND.RES_V3(2).EQ.CBESTV(2)
+ .AND.RES_V3(3).EQ.CBESTV(3).AND.RES_V3(4).LE.CBESTV(4)) THEN
        DUMMY=DUMMY+1
ENDIF

IF(DUMMY.GE.1) THEN
        DO I=1,NRESPONS
                CBESTV(I)=RES_V3(I)
        ENDDO

        DO I=1,NPART
                DO K=1,GMAX
                        COPT_X(I,K)=X3(I,K)
                ENDDO
        ENDDO

        DO J=1,NMACH
                DO K=1,GMAX
                        COPT_Y(J,K)=Y3(J,K)
                ENDDO
        ENDDO

        DO K=1,GMAX
                COPT_Z(K)=Z3(K)
                C_CELLCA(K)=CELLCAP3(K)
                C_CL(K)=CL3(K)
        ENDDO

        DO K=1,GMAX
                DO RE=1,NRE
                        C_RECAP(K,RE)=RECAP3(K,RE)
                        C_TOTCRE(K,RE)=TOTCRE3(K,RE)
                ENDDO
        ENDDO
ENDIF

DUMMY=0
IF(RES_V4(1).LT.CBESTV(1)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V4(1).EQ.CBESTV(1).AND.RES_V4(2).LT.CBESTV(2)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V4(1).EQ.CBESTV(1).AND.RES_V4(2).EQ.CBESTV(2)
+ .AND.RES_V3(3).LT.CBESTV(3)) THEN
        DUMMY=DUMMY+1
ELSEIF(RES_V4(1).EQ.CBESTV(1).AND.RES_V4(2).EQ.CBESTV(2)
+.AND.RES_V4(3).EQ.CBESTV(3).AND.RES_V4(4).LE.CBESTV(4)) THEN
        DUMMY=DUMMY+1
ENDIF
```

```
            IF(DUMMY.GE.1) THEN
                  DO I=1,NRESPONS
                        CBESTV(I)=RES_V4(I)
                  ENDDO

                  DO I=1,NPART
                        DO K=1,GMAX
                              COPT_X(I,K)=X4(I,K)
                        ENDDO
                  ENDDO

                  DO J=1,NMACH
                        DO K=1,GMAX
                              COPT_Y(J,K)=Y4(J,K)
                        ENDDO
                  ENDDO

                  DO K=1,GMAX
                        COPT_Z(K)=Z4(K)
                        C_CELLCA(K)=CELLCAP4(K)
                        C_CL(K)=CL4(K)
                  ENDDO

                  DO K=1,GMAX
                        DO RE=1,NRE
                              C_RECAP(K,RE)=RECAP4(K,RE)
                              C_TOTCRE(K,RE)=TOTCRE4(K,RE)
                        ENDDO
                  ENDDO
            ENDIF
C
C     UPDATING THE BEST SOLUTION VECTOR
C

            DUMMY=0
            IF(CBESTV(1).LT.BEST_V(1)) THEN
                  DUMMY=DUMMY+1
            ELSEIF(CBESTV(1).EQ.BEST_V(1).AND.CBESTV(2).LT.BEST_V(2)) THEN
                  DUMMY=DUMMY+1
            ELSEIF(CBESTV(1).EQ.BEST_V(1).AND.CBESTV(2).EQ.BEST_V(2)
            + .AND.CBESTV(3).LT.BEST_V(3)) THEN
                  DUMMY=DUMMY+1
            ELSEIF(CBESTV(1).EQ.BEST_V(1).AND.CBESTV(2).EQ.BEST_V(2)
            + .AND.CBESTV(3).EQ.BEST_V(3).AND.CBESTV(4).LE.BEST_V(4)) THEN
                  DUMMY=DUMMY+1
            ENDIF

            IF(DUMMY.GE.1) THEN
                  DO I=1,NRESPONS
                        BEST_V(I)=CBESTV(I)
                  ENDDO

                  DO I=1,NPART
                        DO K=1,GMAX
                              XOPT(I,K)=COPT_X(I,K)
                        ENDDO
                  ENDDO

                  DO J=1,NMACH
                        DO K=1,GMAX
                              YOPT(J,K)=COPT_Y(J,K)
```

```
                ENDDO
            ENDDO

            DO K=1,GMAX
                ZOPT(K)=COPT_Z(K)
                O_CELLCA(K)=C_CELLCA(K)
                O_CL(K)=C_CL(K)
            ENDDO

            DO K=1,GMAX
                DO RE=1,NRE
                    O_RECAP(K,RE)=C_RECAP(K,RE)
                    O_TOTCRE(K,RE)=C_TOTCRE(K,RE)
                ENDDO
            ENDDO
        ENDIF

        WRITE(*,*)(BEST_V(I),I=1,NRESPONS)

C
C   UPDATING THE TABU LIST
C

        IF(COUNTER.LE.TABUSIZE) THEN
            DO I=1,NPART
                DO K=1,GMAX
                    TALIST_P( 1,((GMAX-1)*(I-1))+((I-1)+K) )=COPT_X(I,K)
                ENDDO
            ENDDO

            DO J=1,NMACH
                DO K=1,GMAX
                    TALIST_M( 1,((GMAX-1)*(J-1))+((J-1)+K) )=COPT_Y(J,K)
                ENDDO
            ENDDO
        ENDIF

        IF(COUNTER.GT.TABUSIZE) THEN
            DO I=1,NPART
                DO K=1,GMAX
                    TALIST_P(MOD(COUNTER,TABUSIZE),
                    + ((GMAX-1)*(I-1))+((I-1)+K) )=COPT_X(I,K)
                ENDDO
            ENDDO

            DO J=1,NMACH
                DO K=1,GMAX
                    TALIST_M(MOD(COUNTER,TABUSIZE),
                    + ((GMAX-1)*(J-1))+((J-1)+K) )=COPT_Y(J,K)
                ENDDO
            ENDDO
        ENDIF

        WRITE(*,*)(BEST_V(I),I=1,NRESPONS)
        WRITE(7,121) BEST_V(1),BEST_V(2),BEST_V(3),BEST_V(4)
121     FORMAT(F10.2,F10.2,F10.2,F10.0)

        IF(COUNTER.GT.N_ITER) THEN
            EXIT
        ENDIF
```

```
ENDDO

CALL OUTPUT(XOPT,YOPT,ZOPT,NPART,NMACH,GMAX,NRE,
+ O_CELLCA,O_RECAP,O_CL,O_TOTCRE)

CLOSE(UNIT=7)

END

C
C   FUNCTION PRINTING THE BEST SOLUTION OBTAINED
C

SUBROUTINE OUTPUT(XOPT,YOPT,ZOPT,NPART,NMACH,GMAX,NRE,O_CELLCA
+ ,O_RECAP,O_CL,O_TOTCRE)

INTEGER XOPT(100,10),YOPT(100,10),ZOPT(10),NPART,NMACH,GMAX
INTEGER RE,I,J,K,NRE,O_TOTCRE(20,50)
REAL O_CL(20),O_CELLCA(20),O_RECAP(20,50)

OPEN(UNIT=77,STATUS='UNKNOWN',FILE='C:\RECON_OUT\OPT_SOL.TXT')

WRITE(77,*)'=============================================================='
WRITE(77,*)'                          MOCACEF 1.0'
WRITE(77,*)'A TABU SEARCH BASED MULTIPLE OBJECTIVE MATHEMATICAL '
WRITE(77,*)'   PROGRAMMING MODEL FOR CONCURRENTLY FORMING     '
WRITE(77,*)'              PART FAMILIES & MACHINE GROUPS       '
WRITE(77,*)'                   VIA RESOURCE ELEMENTS           '
WRITE(77,*)'                                                   '
WRITE(77,*)'                                                   '
WRITE(77,*)'                    By: Adil BAYKASOGLU            '
WRITE(77,*)'                                                   '
WRITE(77,*)'                    University of Nottingham       '
WRITE(77,*)'          Dept. of Manufacturing Eng. & Operations Mang.'
WRITE(77,*)'                          1998                     '
WRITE(77,*)
WRITE(77,*)'=============================================================='

C  WRITE(77,*) 'OPTIMUM OBJ. FUNC. VALUE:',BOBJ
WRITE(77,*)
WRITE(77,40) GMAX
40    FORMAT('NUMBER OF PRODUCTION CELLS:',I2)
WRITE(77,*)
WRITE(77,*) '================'
WRITE(77,*) 'PART FAMILIES'
WRITE(77,*) '================'
WRITE(77,*)

DO K=1,GMAX
WRITE(77,41) K
41          FORMAT('CELL NUMBER->',I2)
WRITE(77,*)'-------------'

          DO I=1,NPART
                    IF(XOPT(I,K).EQ.1) THEN
                              WRITE(77,50) I
50                            FORMAT('PART-',I2)
                    ENDIF
          ENDDO
WRITE(77,*)
ENDDO
```

```
WRITE(77,*) '=================='
WRITE(77,*) 'MACHINE GROUPS'
WRITE(77,*) '=================='
WRITE(77,*)

DO K=1,GMAX
        WRITE(77,42) K
42      FORMAT('CELL NUMBER->',I2)
        WRITE(77,*)'----------'
        WRITE(77,61) O_CL(K)
        WRITE(77,*)

        IF(O_CELLCA(K).LT.0)THEN
                WRITE(77,62) O_CELLCA(K)
                WRITE(77,*)
        ENDIF

        IF(O_CELLCA(K).GE.0)THEN
                WRITE(77,63) O_CELLCA(K)
                WRITE(77,*)
        ENDIF

61      FORMAT('Cell Capacity Utilisation is = ',F10.2)
62      FORMAT( F10.2,' Units of EXTRA Capacity is Required
        + (i.e. CELL IS BOTTLENECK)')
63      FORMAT( F10.2,' Units of EXTRA Capacity is Available')

        DO J=1,NMACH
                IF(YOPT(J,K).EQ.1) THEN
                        WRITE(77,65) J
65                      FORMAT('MACHINE-',I2)
                ENDIF
        ENDDO
        WRITE(77,*)
ENDDO

WRITE(77,*) '=========================================================='
WRITE(77,*) 'DISTRIBUTION OF RESOURCE ELEMENTS BETWEEN CELLS'
WRITE(77,*) '=========================================================='
WRITE(77,*)

DO K=1,GMAX
        WRITE(77,43)K
43      FORMAT('CELL NUMBER->',I2)
        WRITE(77,*)'----------'
        DO RE=1,NRE
                IF (O_TOTCRE(K,RE).NE.0) THEN
                        WRITE(77,70) O_TOTCRE(K,RE),RE
70                      FORMAT(I2,' Copies of RE-',I2)
                ENDIF
                IF(O_RECAP(K,RE).LT.0) THEN
                        WRITE(77,71) -1*O_RECAP(K,RE),RE
71                      FORMAT( F10.2,' Units of EXTRA Capacity is Required for RE-',I2)
                ENDIF
        ENDDO
        WRITE(77,*)
ENDDO

CLOSE(UNIT=77)
```

```
RETURN
END


C
C  FUNCTION FOR DETERMINATION OF OVERALL DISSIMILARITY MATRIX
C

SUBROUTINE DISSIMIL(P,NPART,NRE,DSM)

INTEGER I,L,P(100,50),NPART,NRE
REAL DSIM(100,100),SDISSIM(100,100),DSM(100,100),W1,W2

OPEN(UNIT=44,STATUS='UNKNOWN',FILE='C:\RECON_OUT\C_DSM.TXT')

CALL PDS(P,NPART,NRE,DSIM)
CALL SDS(NPART,SDISSIM)

W1=0.5
W2=0.5

DO I=1,NPART
        DO L=1,NPART
                DSM(I,L)=W1*DSIM(I,L)+W2*SDISSIM(I,L)
        ENDDO
ENDDO

DO I=2,NPART
        WRITE(44,41)(DSM(I,L),L=1,I-1)
41      FORMAT(20F5.2)
ENDDO

CLOSE(UNIT=44)

RETURN
END
C
C  FUNCTION FOR CALCULATION OF PRODUCTION REQUIREMENT
C  BASED DISSIMILARITY COEFFICIENTS BETWEEN PARTS
C

SUBROUTINE PDS(P,NPART,NRE,DSIM)

INTEGER I,L,RE,ITRS,UNIO,NPART,NRE,P(100,50)
REAL DSIM(100,100)

OPEN(UNIT=10,STATUS='UNKNOWN',FILE='C:\RECON_OUT\PDSM.TXT')

DO I=1,NPART-1
        DO L=I+1,NPART
                ITRS=0
                UNIO=0
                DO RE=1,NRE
                        IF(P(I,RE).NE.0.AND.P(L,RE).NE.0.AND.
                        +P(I,RE).EQ.P(L,RE)) THEN
                                ITRS=ITRS+1
                        ENDIF
                        IF(P(I,RE).NE.0.OR.P(L,RE).NE.0) THEN
                                UNIO=UNIO+1
                        ENDIF
                ENDDO
```

```
                DSIM(I,L)=1-(REAL (ITRS)/(UNIO))
                DSIM(L,I)=DSIM(I,L)
                DSIM(I,I)=0
                DSIM(L,L)=0
        ENDDO
ENDDO

DO I=2,NPART
        WRITE(10,20)(DSIM(I,L),L=1,I-1)
20      FORMAT(20F5.2)
ENDDO

CLOSE(UNIT=10)

RETURN
END

C
C  FUNCTION FOR CALCULATION OF OPERATION SEQUENCE BASED
C  DISSIMILARITY COEFFICIENTS BETWEEN PARTS
C

SUBROUTINE SDS(NPART,SDISSIM)

INTEGER I,J,I1,I2,L1,L2,NPART,SQM(100,100),SDSM(100,100),NOP(100)
INTEGER DISSIM(100,100),ADD,DELETE,SUBSTI,BIG_VAL
REAL SDISSIM(100,100)

OPEN(UNIT=20,STATUS='OLD',FILE='C:\RECON_IN\PRO_SEQ.TXT')
OPEN(UNIT=21,STATUS='UNKNOWN',FILE='C:\RECON_OUT\SDISSIM.TXT')

DO I=1,NPART
        READ(20,*) NOP(I)
ENDDO

DO I=1,NPART
        READ(20,*) (SQM(I,J),J=1,NOP(I))
ENDDO

C
C       CALCULATION OF DISSIMILARITY COEFFICIENTS
C

DO I=1,NPART-1
        DO L=I+1,NPART
                DO I1=1,NOP(I)+1
                        SDSM(I1,1)=I1-1
                ENDDO
                DO L1=1,NOP(L)+1
                        SDSM(1,L1)=L1-1
                ENDDO
                DO I2=2,NOP(I)+1
                        DO L2=2,NOP(L)+1
                                IF(SQM(I,I2-1).EQ.SQM(L,L2-1)) THEN
                                        SUBSTI=SDSM(I2-1,L2-1)
                                ELSE
                                        SUBSTI=SDSM(I2-1,L2-1)+1
                                        DELETE=SDSM(I2-1,L2)+1
                                        ADD=SDSM(I2,L2-1)+1
```

```
                                ENDIF
                                SDSM(I2,L2)=MIN(SUBSTI,DELETE,ADD)
                                DISSIM(I,L)=SDSM(I2,L2)
                        ENDDO
                ENDDO
                DISSIM(L,I)=DISSIM(I,L)
                DISSIM(I,I)=0
                DISSIM(L,L)=0
        ENDDO
ENDDO

BIG_VAL=0
DO I=2,NPART
        DO L=1,I-1
                IF(DISSIM(I,L).GT.BIG_VAL) THEN
                        BIG_VAL=DISSIM(I,L)
                ENDIF
        ENDDO
ENDDO

DO I=1,NPART-1
        DO L=I+1,NPART
                SDISSIM(I,L)=REAL (DISSIM(I,L))/BIG_VAL
                SDISSIM(L,I)=SDISSIM(I,L)
                SDISSIM(I,I)=0
                SDISSIM(L,L)=0
        ENDDO
ENDDO


DO I=2,NPART
        WRITE(21,50)(SDISSIM(I,L),L=1,I-1)
50      FORMAT(20F5.2)
ENDDO

CLOSE(UNIT=20)
CLOSE(UNIT=21)

RETURN
END

C
C  FUNCTION FOR CALCULATION OF OBJECTIVE AND CONSTRAINT FUNCTIONS
C

SUBROUTINE OBJFUNC(X,Y,Z,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
+ MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
+ OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

INTEGER I,K,L,J,RE,NRE,II,JJ,Z(10),X(100,10),Y(100,10)
INTEGER XTOP,YTOP,TOPYZ1,TOPYZ2,GXYITOP,MC2(100),P2(100)
INTEGER TOPXZ1,TOPXZ2,TOTAL1,TOTAL2,NPART,NMACH,GMAX
INTEGER R,GXTOP,GYTOP,GTOTAL1,GTOTAL2,GXYZTOP,MMIN,ICM
INTEGER MMAX,PMIN,PMAX,GTOTAL3,GTOTAL4,TOTAL3,TOTAL4
INTEGER P(100,50),M(100,50),P1(100,50),M1(100,50),INCELLM
INTEGER P_QAN(100),FEMIX(20,50),REPRE,TOTCRE(20,50)
REAL M_CAP(50),P_MT(100,100),CAP_RE
REAL CAP_FAC(20),CAP_AV,CAP_RE1,CAP_AV1,CAP_FAC1(20,50)
REAL DSM(100,100),CL(20),ACL,TOTUTL,OBJ1,OBJ2,OBJ3,OBJ4,OBJ5
REAL OBJ2_3
```

```
C
C  EQUALITY CONSTRAINTS
C

C
C     CONSTRAINT-1
C

GXTOP=0
DO I=1,NPART
        XTOP=0
        DO K=1,GMAX
                XTOP=XTOP+(X(I,K)*Z(K))
        ENDDO
        GXTOP=GXTOP+XTOP-1
ENDDO

C
C     CONSTRAINT-2
C

GYTOP=0
DO J=1,NMACH
        YTOP=0
        DO K=1,GMAX
                YTOP=YTOP+(Y(J,K)*Z(K))
        ENDDO
        GYTOP=GYTOP+YTOP-1
ENDDO

C
C     CONSTRAINT-3
C

INCELLM=0
DO K=1,GMAX
        DO I=1,NMACH
                DO RE=1,NRE
                        M1(I,RE)=0
                ENDDO
        ENDDO
        DO J=1,NPART
                DO RE=1,NRE
                        P1(J,RE)=0
                ENDDO
        ENDDO
        DO RE=1,NRE
                MC2(RE)=0
        ENDDO
        DO RE=1,NRE
                P2(RE)=0
        ENDDO
        DO I=1,NPART
                IF(X(I,K).EQ.1) THEN
                DO RE=1,NRE
                        P1(I,RE)=P(I,RE)
                ENDDO
                DO II=1,NPART
                        DO RE=1,NRE
                                IF(P1(1,RE).EQ.0.AND.P1(II+1,RE).EQ.0.AND.P2(RE).EQ.0)
                                +THEN
```

```
                                P2(RE)=0
                        ELSE
                                P2(RE)=1
                        ENDIF
                ENDDO
        ENDDO
        ENDIF
ENDDO
DO J=1,NMACH
        IF(Y(J,K).EQ.1) THEN
        DO RE=1,NRE
                M1(J,RE)=M(J,RE)
        ENDDO
        DO JJ=1,NMACH
                DO RE=1,NRE
                        IF(M1(1,RE).EQ.0.AND.M1(JJ+1,RE).EQ.0.
                        +AND.MC2(RE).EQ.0) THEN
                                MC2(RE)=0
                        ELSE
                                MC2(RE)=1
                        ENDIF
                ENDDO
        ENDDO
        ENDIF
ENDDO

ICM=0
DO RE=1,NRE
        IF(P2(RE).EQ.1.AND.MC2(RE).EQ.0) THEN
                ICM=ICM+1
        ENDIF
        ENDDO

        INCELLM=INCELLM+ICM
ENDDO

GXYITOP=ABS(GXTOP)+ABS(GYTOP)+ABS(INCELLM)

C
C  NON-EQUALITY CONSTRAINTS
C

C
C  CONSTRAINT-4
C

GTOTAL2=0
 DO K=1,GMAX
        TOTAL2=0
        TOPXZ1=0
        DO I=1,NPART
                TOPXZ1=TOPXZ1+X(I,K)
        ENDDO
        TOTAL2=TOPXZ1-PMIN*Z(K)
        IF(TOTAL2.LT.0) THEN
                R=1
        ELSE
                R=0
        ENDIF
        GTOTAL2=GTOTAL2+R*TOTAL2
ENDDO
```

```
C
C    CONSTRAINT-5
C

GTOTAL1=0
DO K=1,GMAX
        TOTAL1=0
        TOPYZ1=0
        DO J=1,NMACH
                TOPYZ1=TOPYZ1+Y(J,K)
        ENDDO
        TOTAL1=TOPYZ1-MMIN*Z(K)
        IF(TOTAL1.LT.0) THEN
                R=1
        ELSE
                R=0
        ENDIF
        GTOTAL1=GTOTAL1+R*TOTAL1
ENDDO

C
C    CONSTRAINT-6
C

GTOTAL4=0
DO K=1,GMAX
        TOTAL4=0
        TOPXZ2=0
        DO I=1,NPART
                TOPXZ2=TOPXZ2+X(I,K)
        ENDDO
        TOTAL4=PMAX*Z(K)-TOPXZ2
        IF(TOTAL4.LT.0) THEN
                R=1
        ELSE
                R=0
        ENDIF
        GTOTAL4=GTOTAL4+R*TOTAL4
ENDDO

C
C    CONSTRAINT-7
C

GTOTAL3=0
DO K=1,GMAX
        TOTAL3=0
        TOPYZ2=0
        DO J=1,NMACH
                TOPYZ2=TOPYZ2+Y(J,K)
        ENDDO
        TOTAL3=MMAX*Z(K)-TOPYZ2
        IF(TOTAL3.LT.0) THEN
                R=1
        ELSE
                R=0
        ENDIF
        GTOTAL3=GTOTAL3+R*TOTAL3
ENDDO
```

```
GXYZTOP=ABS(GTOTAL1)+ABS(GTOTAL2)+ABS(GTOTAL3)+ABS(GTOTAL4)

C
C    CALCULATION OF CELL UTILISATIONS
C

DO K=1,GMAX
        CAP_AV=0
        DO J=1,NMACH
                CAP_AV=CAP_AV+M_CAP(J)*Y(J,K)
        ENDDO
        CAP_RE=0
        DO I=1,NPART
                DO RE=1,NRE
                        CAP_RE=CAP_RE+X(I,K)*P_MT(I,RE)*P_QAN(I)
                ENDDO
        ENDDO

        IF(CAP_AV.NE.0) THEN
                CL(K)=CAP_RE/CAP_AV
        ELSE
                CL(K)=0
        ENDIF
ENDDO

C
C    CALCULATION OF AVERAGE UTILISATION OF CELLS
C

TOTUTL=0
DO K=1,GMAX
        TOTUTL=TOTUTL+CL(K)
ENDDO

ACL=TOTUTL/REAL(GMAX)

C
C    DETERMINATION OF FLEXIBILITY MATRIX TOTCRE
C

DO K=1,GMAX
        DO RE=1,NRE
                REPRE=0
                DO J=1,NMACH
                        REPRE=REPRE+(Y(J,K)*M(J,RE))
                ENDDO
                TOTCRE(K,RE)=REPRE
        ENDDO
ENDDO


C
C  CALCULATION OF OBJECTIVE FUNCTION-1 (i.e. PART SIMILARITY)
C

OBJ1=0
DO K=1,GMAX
        DO I=1,NPART
                DO L=1,NPART
                        OBJ1=OBJ1+(DSM(I,L)*0.5*X(I,K)*X(L,K))
                ENDDO
```

```
        ENDDO
ENDDO

C
C    OBJECTIVE-2
C

DO K=1,GMAX
        CAP_AV=0
        DO J=1,NMACH
                CAP_AV=CAP_AV+M_CAP(J)*Y(J,K)
        ENDDO
        CAP_RE=0
        DO I=1,NPART
                DO RE=1,NRE
                        CAP_RE=CAP_RE+X(I,K)*P_MT(I,RE)*P_QAN(I)
                ENDDO
        ENDDO
        CAP_FAC(K)=CAP_AV-CAP_RE
ENDDO
OBJ2=0
DO K=1,GMAX
        IF (CAP_FAC(K).LT.0) THEN
                OBJ2=OBJ2+ABS(CAP_FAC(K))
        ENDIF
ENDDO

C
C  OBJECTIVE-3
C

DO K=1,GMAX
        DO RE=1,NRE
                CAP_AV1=0
                DO J=1,NMACH
                        CAP_AV1=CAP_AV1+M_CAP(J)*Y(J,K)*M(J,RE)
                ENDDO
                CAP_RE1=0
                DO I=1,NPART
                        CAP_RE1=CAP_RE1+X(I,K)*P_MT(I,RE)*P_QAN(I)
                ENDDO
                CAP_FAC1(K,RE)=CAP_AV1-CAP_RE1
        ENDDO
ENDDO

OBJ3=0
DO K=1,GMAX
        DO RE=1,NRE
                IF(CAP_FAC1(K,RE).LT.0) THEN
                        OBJ3=OBJ3+ABS(CAP_FAC1(K,RE))
                ENDIF
        ENDDO
ENDDO

OBJ2_3=OBJ2+OBJ3

C
C OBJECTIVE -4
C

OBJ4=0
```

```
DO K=1,GMAX
        OBJ4=OBJ4+( (CL(K)-ACL)**2 /GMAX )
ENDDO

C
C   CALCULATION OF OBJECTIVE FUNCTION-5
C

OBJ5=0
DO K=1,GMAX
        DO RE=1,NRE
                IF(TOTCRE(K,RE).NE.0) THEN
                        OBJ5=OBJ5+TOTCRE(K,RE)/TOTCRE(K,RE)
                ENDIF
        ENDDO
ENDDO

RETURN
END


C
C  FUNCTION FOR GENERATING THE
C  INITIAL FEASIBLE RANDOM SOLUTION
C

SUBROUTINEINRANSOL(ISEED,GMAX,NPART,NMACH,P,M,DSM,NRE,
+ MMIN,MMAX,PMIN,PMAX,M_CAP,P_MT,P_QAN,X,Y,Z)

INTEGER I,J,K,X(100,10),Y(100,10),Z(10),PMIN,PMAX,MMIN,MMAX,NRE
INTEGER ISEED,GMAX,NPART,NMACH,P_QAN(100),P(100,50),M(100,50)
INTEGER GXYITOP,GXYZTOP,TOTCRE(20,50)
REAL P_MT(100,100),M_CAP(50),DSM(100,100),OBJ1,OBJ2_3,OBJ4,OBJ5
REAL CL(20),CAP_FAC(20),CAP_FAC1(20,50)

DO
        DO I=1,NPART
                DO K=1,GMAX
                        X(I,K)=0
                ENDDO
                K=NINT(ran(iseed)*(GMAX-1)+1)
                X(I,K)=1
        ENDDO

        DO J=1,NMACH
                DO K=1,GMAX
                        Y(J,K)=0
                ENDDO
                K=NINT(ran(iseed)*(GMAX-1)+1)
                Y(J,K)=1
        ENDDO

        DO K=1,GMAX
                Z(K)=1
        ENDDO

        CALL OBJFUNC(X,Y,Z,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
        +  MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,OBJ1,
        + OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

        IF(GXYITOP.EQ.0.AND.GXYZTOP.EQ.0)THEN
```

```
                EXIT
        ENDIF
ENDDO

RETURN
END

C
C  FUNCTION FOR GENERATING THE INITIAL FEASIBLE RANDOM SOLUTION
C

SUBROUTINE INFESS(GMAX,X,Y,Z)

INTEGER I,J,K,X(100,10),Y(100,10),Z(10),GMAX
OPEN(UNIT=2,STATUS='OLD',FILE='C:\RECON_IN\incelld.TXT')

READ(2,*)NPART,NMACH
DO I=1,NPART
        READ(2,*) (X(I,K),K=1,GMAX)
ENDDO

DO J=1,NMACH
        READ(2,*) (Y(J,K),K=1,GMAX)
ENDDO

DO K=1,GMAX
        READ(2,*) Z(K)
ENDDO

CLOSE(UNIT=2)
RETURN
END

C
C  FUNCTION FOR CREATING FEASIBLE, NON-TABU NEIGHBOUR SOLUTIONS
C

SUBROUTINE MOVEMENT(ISEED,GMAX,NPART,NMACH,P,M,DSM,NRE,MMIN,
+ MMAX,PMIN,PMAX,M_CAP,P_MT,P_QAN,TABUSIZE,TALIST_P,TALIST_M
+  ,X,Y,Z,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)

INTEGER I,J,K,Z1(10),X1(100,10),Y1(100,10)
INTEGER Z(10),X(100,10),Y(100,10),TOTCRE(20,50)
INTEGER Z2(10),X2(100,10),Y2(100,10),Z4(10),X4(100,10),Y4(100,10)
INTEGER Z3(10),X3(100,10),Y3(100,10)
INTEGER A,B,AA,BB,SAME1,SAME2,SIM_TV(50),SIM_TW(50)
INTEGER SIM_NV(20),SIM_NW(20),SAMESV_M(50,500),SAMESV_P(50,500)
INTEGER GMAX,NPART,NMACH,NRE,MMIN,MMAX,PMIN,PMAX,ISEED
INTEGER P(100,50),M(100,50),P_QAN(100),TALIST_P(40,100) ,TABUSIZE
INTEGER TALIST_M(40,100),GXYITOP,GXYZTOP,DUMMY
REAL M_CAP(50),P_MT(100,100),DSM(100,100),OBJ1,OBJ2_3,OBJ4,OBJ5
REAL CL(20),CAP_FAC(20),CAP_FAC1(20,50),COBJ1,COBJ2_3,COBJ4,COBJ5

CALL OBJFUNC(X,Y,Z,P,M,DSM,NMACH,NPART,NRE,MMIN,GMAX,
+MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,GXYZTOP,
+COBJ1,COBJ2_3,COBJ4,COBJ5,CL,CAP_FAC,CAP_FAC1)

C     First vector
DO
```

```
C       Initialisation

        DO I=1,NPART
                DO K=1,GMAX
                        X1(I,K)=X(I,K)
                ENDDO
        ENDDO

        DO J=1,NPART
                DO K=1,GMAX
                        Y1(J,K)=Y(J,K)
                ENDDO
        ENDDO

        DO K=1,GMAX
                Z1(K)=Z(K)
        ENDDO

C   Changing X1(I,K)

        I=NINT(ran(iseed)*(NPART-1)+1)
        DO K=1,GMAX
                IF(X1(I,K).EQ.1) THEN
                        X1(I,K)=0
                ENDIF
        ENDDO
        K=NINT(ran(iseed)*(GMAX-1)+1)
        X1(I,K)=1

C   Changing Y1(J,K)
        J=NINT(ran(iseed)*(NMACH-1)+1)
        DO K=1,GMAX
                IF(Y1(J,K).EQ.1) THEN
                        Y1(J,K)=0
                ENDIF
        ENDDO
        K=NINT(ran(iseed)*(GMAX-1)+1)
        Y1(J,K)=1

        CALL OBJFUNC(X1,Y1,Z1,P,M,DSM,NMACH,NPART,NRE,MMIN,
        + GMAX, MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,
        +GXYZTOP,OBJ1, OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

        DO I=1,NPART
                DO K=1,GMAX
                        SAMESV_P( 1,((GMAX-1)*(I-1))+((I-1)+K) )=X1(I,K)
                ENDDO
        ENDDO
        DO J=1,NMACH
                DO K=1,GMAX
                        SAMESV_M( 1,((GMAX-1)*(J-1))+((J-1)+K) )=Y1(J,K)
                ENDDO
        ENDDO

C   Checking tabu list
        DO K=1,TABUSIZE
                SIM_TV(K)=0
                SIM_TW(K)=0
        ENDDO

        DO K=1,TABUSIZE
```

```
            AA=0
            DO J=1,GMAX*NMACH
                  AA=AA+ABS(SAMESV_M(1,J)-TALIST_M(K,J))
            ENDDO
            SIM_TV(K)=AA
            BB=0
            DO I=1,GMAX*NPART
                  BB=BB+ABS(SAMESV_P(1,I)-TALIST_P(K,I))
            ENDDO
            SIM_TW(K)=BB
ENDDO

SAME2=0
DO K=1,TABUSIZE
            IF(SIM_TV(K).EQ.0.AND.SIM_TW(K).EQ.0) THEN
                  SAME2=SAME2+1
            ENDIF
ENDDO

DUMMY=0
IF(OBJ1.LT.COBJ1) THEN
            DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.LT.COBJ2_3) THEN
            DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+.AND.OBJ4.LT.COBJ4) THEN
            DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+.AND.OBJ4.EQ.COBJ4.AND.OBJ5.LE.COBJ5) THEN
            DUMMY=DUMMY+1
ENDIF
            IF(SAME2.EQ.0.AND.GXYITOP.EQ.0.AND.GXYZTOP.EQ.0) THEN
                  EXIT
            ELSEIF(DUMMY.GE.1.AND.GXYITOP.EQ.0.AND.GXYZTOP.EQ.0) THEN
                  EXIT
            ENDIF
ENDDO

C     Second vector
DO
C      Initialisation

            DO I=1,NPART
                  DO K=1,GMAX
                        X2(I,K)=X(I,K)
                  ENDDO
            ENDDO
            DO J=1,NPART
                  DO K=1,GMAX
                        Y2(J,K)=Y(J,K)
                  ENDDO
            ENDDO
            DO K=1,GMAX
                  Z2(K)=Z(K)
            ENDDO

C     Changing X2(I,K)
            I=NINT(ran(iseed)*(NPART-1)+1)
            DO K=1,GMAX
                  IF(X2(I,K).EQ.1) THEN
```

```
                    X2(I,K)=0
                ENDIF
            ENDDO
            K=NINT(ran(iseed)*(GMAX-1)+1)
            X2(I,K)=1

C   Changing Y2(J,K)
            J=NINT(ran(iseed)*(NMACH-1)+1)
            DO K=1,GMAX
                IF(Y2(J,K).EQ.1) THEN
                    Y2(J,K)=0
                ENDIF
            ENDDO
            K=NINT(ran(iseed)*(GMAX-1)+1)
            Y2(J,K)=1

            CALL OBJFUNC(X2,Y2,Z2,P,M,DSM,NMACH,NPART,NRE,MMIN,
            +GMAX, MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYTOP,
            +GXYZTOP,OBJ1,OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

            DO I=1,NPART
                DO K=1,GMAX
                    SAMESV_P( 2,((GMAX-1)*(I-1))+((I-1)+K) )=X2(I,K)
                ENDDO
            ENDDO
            DO J=1,NMACH
                DO K=1,GMAX
                    SAMESV_M( 2,((GMAX-1)*(J-1))+((J-1)+K) )=Y2(J,K)
                ENDDO
            ENDDO

C   Checking same solution vectors
            SIM_NV(1)=0
            SIM_NW(1)=0

            A=0
            DO J=1,GMAX*NMACH
                A=A+ABS(SAMESV_M(1,J)-SAMESV_M(2,J))
            ENDDO
            SIM_NV(1)=A

            B=0
            DO I=1,GMAX*NPART
                B=B+ABS(SAMESV_P(1,I)-SAMESV_P(2,I))
            ENDDO
            SIM_NW(1)=B

            SAME1=0
            IF(SIM_NV(1).EQ.0.AND.SIM_NW(1).EQ.0) THEN
                SAME1=SAME1+1
            ENDIF

C   Checking tabu list
            DO K=1,TABUSIZE
                SIM_TV(K)=0
                SIM_TW(K)=0
            ENDDO

            DO K=1,TABUSIZE
                AA=0
                DO J=1,GMAX*NMACH
```

```
                AA=AA+ABS(SAMESV_M(2,J)-TALIST_M(K,J))
        ENDDO
        SIM_TV(K)=AA
        BB=0
        DO I=1,GMAX*NPART
                BB=BB+ABS(SAMESV_P(2,I)-TALIST_P(K,I))
        ENDDO
        SIM_TW(K)=BB
ENDDO

SAME2=0
DO K=1,TABUSIZE
        IF(SIM_TV(K).EQ.0.AND.SIM_TW(K).EQ.0) THEN
                SAME2=SAME2+1
        ENDIF
ENDDO

DUMMY=0
IF(OBJ1.LT.COBJ1) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.LT.COBJ2_3) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+ .AND.OBJ4.LT.COBJ4) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+ .AND.OBJ4.EQ.COBJ4.AND.OBJ5.LE.COBJ5) THEN
        DUMMY=DUMMY+1
ENDIF

IF(SAME1.EQ.0.AND.SAME2.EQ.0.AND.GXYITOP.EQ.0.AND.GXYZTOP.EQ.0)
        + THEN
                EXIT
        ELSEIF(DUMMY.GE.1.AND.SAME1.EQ.0.AND.GXYITOP.EQ.0.AND.
        + GXYZTOP.EQ.0) THEN
                EXIT
        ENDIF
ENDDO

C     Third vector
DO
C     Initialisation

        DO I=1,NPART
                DO K=1,GMAX
                        X3(I,K)=X(I,K)
                ENDDO
        ENDDO
        DO J=1,NPART
                DO K=1,GMAX
                        Y3(J,K)=Y(J,K)
                ENDDO
        ENDDO
        DO K=1,GMAX
                Z3(K)=Z(K)
        ENDDO

C   Changing X3(I,K)
        I=NINT(ran(iseed)*(NPART-1)+1)
        DO K=1,GMAX
                IF(X3(I,K).EQ.1) THEN
```

```
                    X3(I,K)=0
                ENDIF
            ENDDO
            K=NINT(ran(iseed)*(GMAX-1)+1)
            X3(I,K)=1

C  Changing Y3(J,K)
            J=NINT(ran(iseed)*(NMACH-1)+1)
            DO K=1,GMAX
                IF(Y3(J,K).EQ.1) THEN
                    Y3(J,K)=0
                ENDIF
            ENDDO
            K=NINT(ran(iseed)*(GMAX-1)+1)
            Y3(J,K)=1

            CALL OBJFUNC(X3,Y3,Z3,P,M,DSM,NMACH,NPART,NRE,MMIN,
            +GMAX,MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,
            +GXYZTOP,OBJ1, OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

            DO I=1,NPART
                DO K=1,GMAX
                    SAMESV_P( 3,((GMAX-1)*(I-1))+((I-1)+K) )=X3(I,K)
                ENDDO
            ENDDO
            DO J=1,NMACH
                DO K=1,GMAX
                    SAMESV_M( 3,((GMAX-1)*(J-1))+((J-1)+K) )=Y3(J,K)
                ENDDO
            ENDDO

C     Checking same solution vectors
            DO K=1,2
                SIM_NV(K)=0
                SIM_NW(K)=0
            ENDDO

            DO K=1,2
                A=0
                DO J=1,GMAX*NMACH
                    A=A+ABS(SAMESV_M(K,J)-SAMESV_M(3,J))
                ENDDO
                SIM_NV(K)=A
                B=0
                DO I=1,GMAX*NPART
                    B=B+ABS(SAMESV_P(K,I)-SAMESV_P(3,I))
                ENDDO
                SIM_NW(K)=B
            ENDDO

            SAME1=0
            DO K=1,2
                IF(SIM_NV(K).EQ.0.AND.SIM_NW(K).EQ.0) THEN
                    SAME1=SAME1+1
                ENDIF
            ENDDO

C     Checking tabu list
            DO K=1,TABUSIZE
                SIM_TV(K)=0
                SIM_TW(K)=0
```

```
ENDDO

DO K=1,TABUSIZE
        AA=0
        DO J=1,GMAX*NMACH
                AA=AA+ABS(SAMESV_M(3,J)-TALIST_M(K,J))
        ENDDO
        SIM_TV(K)=AA
        BB=0
        DO I=1,GMAX*NPART
                BB=BB+ABS(SAMESV_P(3,I)-TALIST_P(K,I))
        ENDDO
        SIM_TW(K)=BB
ENDDO

SAME2=0
DO K=1,TABUSIZE
        IF(SIM_TV(K).EQ.0.AND.SIM_TW(K).EQ.0) THEN
                SAME2=SAME2+1
        ENDIF
ENDDO

DUMMY=0
IF(OBJ1.LT.COBJ1) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.LT.COBJ2_3) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+ .AND.OBJ4.LT.COBJ4) THEN
        DUMMY=DUMMY+1
ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
+ .AND.OBJ4.EQ.COBJ4.AND.OBJ5.LE.COBJ5) THEN
        DUMMY=DUMMY+1
ENDIF

IF(SAME1.EQ.0.AND.SAME2.EQ.0.AND.GXYITOP.EQ.0.AND.GXYZTOP.EQ.0)
+  THEN
        EXIT
ELSEIF(DUMMY.GE.1.AND.SAME1.EQ.0.AND.GXYITOP.EQ.0.AND.
+ GXYZTOP.EQ.0) THEN
        EXIT
ENDIF
ENDDO

C   Fourth vector
DO
C   Initialisation
        DO I=1,NPART
                DO K=1,GMAX
                        X4(I,K)=X(I,K)
                ENDDO
        ENDDO
        DO J=1,NPART
                DO K=1,GMAX
                        Y4(J,K)=Y(J,K)
                ENDDO
        ENDDO
        DO K=1,GMAX
                Z4(K)=Z(K)
        ENDDO
```

```
C   Changing X4(I,K)
        I=NINT(ran(iseed)*(NPART-1)+1)
        DO K=1,GMAX
                IF(X4(I,K).EQ.1) THEN
                        X4(I,K)=0
                ENDIF
        ENDDO
        K=NINT(ran(iseed)*(GMAX-1)+1)
        X4(I,K)=1

C  Changing Y4(J,K)
         J=NINT(ran(iseed)*(NMACH-1)+1)
         DO K=1,GMAX
                IF(Y4(J,K).EQ.1) THEN
                        Y4(J,K)=0
                ENDIF
         ENDDO
          K=NINT(ran(iseed)*(GMAX-1)+1)
          Y4(J,K)=1

        CALL OBJFUNC(X4,Y4,Z4,P,M,DSM,NMACH,NPART,NRE,MMIN,
        +GMAX,MMAX,PMAX,PMIN,M_CAP,P_MT,P_QAN,TOTCRE,GXYITOP,
        +GXYZTOP,OBJ1,OBJ2_3,OBJ4,OBJ5,CL,CAP_FAC,CAP_FAC1)

        DO I=1,NPART
                DO K=1,GMAX
                        SAMESV_P( 4,((GMAX-1)*(I-1))+((I-1)+K) )=X4(I,K)
                ENDDO
        ENDDO
        DO J=1,NMACH
                DO K=1,GMAX
                        SAMESV_M( 4,((GMAX-1)*(J-1))+((J-1)+K) )=Y4(J,K)
                ENDDO
        ENDDO

C   Checking same solution vectors
        DO K=1,2
                SIM_NV(K)=0
                SIM_NW(K)=0
        ENDDO

        DO K=1,3
                A=0
                DO J=1,GMAX*NMACH
                        A=A+ABS(SAMESV_M(K,J)-SAMESV_M(4,J))
                ENDDO
                SIM_NV(K)=A
                B=0
                DO I=1,GMAX*NPART
                        B=B+ABS(SAMESV_P(K,I)-SAMESV_P(4,I))
                ENDDO
                SIM_NW(K)=B
        ENDDO

        SAME1=0
        DO K=1,3
                IF(SIM_NV(K).EQ.0.AND.SIM_NW(K).EQ.0) THEN
                        SAME1=SAME1+1
                ENDIF
        ENDDO
```

```
C   Checking tabu list
        DO K=1,TABUSIZE
            SIM_TV(K)=0
            SIM_TW(K)=0
        ENDDO

        DO K=1,TABUSIZE
            AA=0
            DO J=1,GMAX*NMACH
                AA=AA+ABS(SAMESV_M(4,J)-TALIST_M(K,J))
            ENDDO
            SIM_TV(K)=AA
            BB=0
            DO I=1,GMAX*NPART
                BB=BB+ABS(SAMESV_P(4,I)-TALIST_P(K,I))
            ENDDO
            SIM_TW(K)=BB

        ENDDO

        SAME2=0
        DO K=1,TABUSIZE
            IF(SIM_TV(K).EQ.0.AND.SIM_TW(K).EQ.0) THEN
                SAME2=SAME2+1
            ENDIF
        ENDDO

        DUMMY=0
        IF(OBJ1.LT.COBJ1) THEN
            DUMMY=DUMMY+1
        ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.LT.COBJ2_3) THEN
            DUMMY=DUMMY+1
        ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
        + .AND.OBJ4.LT.COBJ4) THEN
            DUMMY=DUMMY+1
        ELSEIF(OBJ1.EQ.COBJ1.AND.OBJ2_3.EQ.COBJ2_3
        +  .AND.OBJ4.EQ.COBJ4.AND.OBJ5.LE.COBJ5) THEN
            DUMMY=DUMMY+1
        ENDIF

        IF(SAME1.EQ.0.AND.SAME2.EQ.0.AND.GXYITOP.EQ.0.AND.GXYZTOP.EQ.0)
        + THEN
            EXIT
        ELSEIF(DUMMY.GE.1.AND.SAME1.EQ.0.AND.GXYITOP.EQ.0.AND.
        + GXYZTOP.EQ.0) THEN
            EXIT
        ENDIF
ENDDO

RETURN
END
```

Appendix V : Application of MOCACEF 1.0

The computer program (MOCACEF 1.0) which was given in Appendix IV is applied to a real job shop that has 22 machines and produces 20 different part types. The data files and computer program output for this case study are given in the following Tables.

Table V.1 Machines and their capabilities based on REs

(MACH_RE.TXT)

Machines	*Resource Elements*
	1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3
	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
Machine-1	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0*
Machine-2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0
Machine-3	0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0
Machine-5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0
Machine-6	0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-7	0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-8	0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-9	1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-10	1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-11	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-12	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-13	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-14	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0
Machine-16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0
Machine-17	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0
Machine-18	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
Machine-19	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
Machine-20	1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-21	1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Machine-22	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

*1: *RE is available from the corresponding machine.*
0: *RE is not available from the corresponding machine*

Table V.2 Annual machine capacity

(MACH_CAP.TXT)

Machines	*Annual capacity(min)*
Machine-1	140000
Machine-2	140000
Machine-3	140000
Machine-4	140000
Machine-5	140000
Machine-6	140000
Machine-7	140000
Machine-8	140000
Machine-9	140000
Machine-10	140000
Machine-11	140000
Machine-12	140000
Machine-13	140000
Machine-14	140000
Machine-15	140000
Machine-16	140000
Machine-17	140000
Machine-18	140000
Machine-19	140000
Machine-20	140000
Machine-21	140000
Machine-22	140000

Table V.3 Parts processing requirements as generic process plans based on REs (JOB_RE.TXT)

Parts	*Resource Elements*
	1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3
	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
Part-1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0
Part-2	0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-3	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
Part-5	1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-6	0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-7	0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-8	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
Part-9	0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-10	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-11	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0
Part-12	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
Part-13	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
Part-14	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
Part-16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
Part-17	0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-18	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1
Part-19	1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Part-20	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1

1: RE is required by the corresponding part,
0: RE is not required by the corresponding part

Table V.4 RE based part processing sequence data (PRO_SEQ.TXT)

Parts	*# Operations based on Res*	*RE based Operation Sequence*
Part-1	3	29 26 30
Part-2	3	7 6 5
Part-3	2	1 5
Part-4	2	17 21
Part-5	3	1 5 2
Part-6	5	10 12 14 8 15
Part-7	4	9 8 11 12
Part-8	3	25 32 26
Part-9	4	10 8 9 12
Part-10	4	16 10 12 14
Part-11	4	17 18 19 21
Part-12	3	17 19 21
Part-13	3	19 21 20
Part-14	2	6 7
Part-15	2	22 21
Part-16	3	29 24 25
Part-17	4	10 8 9 12
Part-18	4	29 24 25 32
Part-19	3	1 5 3
Part-20	2	26 32

Table V.5 Part processing time and annual demand

(PRO_TIME.TXT)

Parts	*Processing + set-up time requirements for Resource Elements*	*Annual Demand*
Part-1	0 0 0

Table V.6 Output of the MOCACEF 1.0
(OPT_SOL.TXT)

NUMBER OF PRODUCTION CELLS: 4

PART FAMILIES

CELL NUMBER-> 1

PART- 2
PART- 3
PART- 5
PART- 14
PART- 19

CELL NUMBER-> 2

PART- 6
PART- 7
PART- 9
PART-10
PART- 17

CELL NUMBER-> 3

PART-4
PART-11
PART-12
PART-13
PART-15

CELL NUMBER-> 4

PART-1
PART-8
PART-16
PART-18
PART-20

MACHINE GROUPS

CELL NUMBER-> 1

Cell Capacity Utilisation is = .88

99060.00 Units of EXTRA Capacity is Available

MACHINE- 8
MACHINE- 9
MACHINE-11
MACHINE-12
MACHINE-19
MACHINE-21

CELL NUMBER-> 2

Cell Capacity Utilisation is = .79

149915.00 Units of EXTRA Capacity is Available

MACHINE- 3
MACHINE- 6
MACHINE-10
MACHINE-13
MACHINE-17

CELL NUMBER-> 3

Cell Capacity Utilisation is = .92

70620.00 Units of EXTRA Capacity is Available

MACHINE- 2
MACHINE- 5
MACHINE- 7
MACHINE-14
MACHINE-16
MACHINE-22

CELL NUMBER-> 4

Cell Capacity Utilisation is = .92

52996.00 Units of EXTRA Capacity is Available

MACHINE- 1
MACHINE- 4
MACHINE-15
MACHINE-18
MACHINE-20

DISTRIBUTION OF RESOURCE ELEMENTS BETWEEN CELLS

CELL NUMBER-> 1
---
2 Copies of RE- 1
2 Copies of RE- 2
1 Copies of RE- 3
2 Copies of RE- 4
2 Copies of RE- 5
8440.00 Units of EXTRA Capacity is Required for RE- 5
2 Copies of RE- 6
2 Copies of RE- 7
1 Copies of RE- 8
1 Copies of RE- 9
1 Copies of RE-11
1 Copies of RE-13
1 Copies of RE-24
1 Copies of RE-25
1 Copies of RE-26
1 Copies of RE-27
1 Copies of RE-28

1 Copies of RE-29
1 Copies of RE-30
1 Copies of RE-31

CELL NUMBER-> 2
---
1 Copies of RE- 1
1 Copies of RE- 2
1 Copies of RE- 4
1 Copies of RE- 6
1 Copies of RE- 7
1 Copies of RE- 8
2 Copies of RE- 9
1 Copies of RE-10
77295.00 Units of EXTRA Capacity is Required for RE-10
1 Copies of RE-11
1 Copies of RE-12
1 Copies of RE-13
1 Copies of RE-14
1 Copies of RE-15
1 Copies of RE-16
1 Copies of RE-25
1 Copies of RE-26
1 Copies of RE-28
1 Copies of RE-29
1 Copies of RE-30
1 Copies of RE-31

CELL NUMBER-> 3
---
1 Copies of RE- 8
1 Copies of RE- 9
1 Copies of RE-11
1 Copies of RE-13
1 Copies of RE-17
2 Copies of RE-18
2 Copies of RE-19
1 Copies of RE-20
1 Copies of RE-21
120100.00 Units of EXTRA Capacity is Required for RE-21
2 Copies of RE-22
2 Copies of RE-23
1 Copies of RE-24
1 Copies of RE-25
1 Copies of RE-26
1 Copies of RE-27
1 Copies of RE-28
1 Copies of RE-29
1 Copies of RE-30
1 Copies of RE-31

CELL NUMBER-> 4
---
1 Copies of RE- 1
1 Copies of RE- 2
1 Copies of RE- 3
1 Copies of RE- 4
1 Copies of RE- 6
1 Copies of RE- 7
1 Copies of RE-18
1 Copies of RE-19

1 Copies of RE-20
1 Copies of RE-22
1 Copies of RE-23
2 Copies of RE-24
2 Copies of RE-25
2 Copies of RE-26
2 Copies of RE-27
2 Copies of RE-28
2 Copies of RE-29
2 Copies of RE-30
2 Copies of RE-31
1 Copies of RE-32

As the above results represent, the program can successfully determine cellular configuration. Unlike the existing models, it can also determine which capability units are required to satisfy the total demand. So, it is possible to invest on machines whose minimum capability requirements are known (i.e. RE-5 in Cell-1, RE-10 in Cell2, RE-21 in Cell-3). Therefore, the number of machine duplications can be reduced.

Appendix VI : Frames for loading and reconfiguration modules

A brief description of functions (subroutines) to implement the loading and reconfiguration modules of multiple objective decision support framework are given in Frames D.1 and D.2 below. Full program listings are around 4000 lines of C/C++ code.

Frame D.1

```
//
//              MULTIPLE OBJECTIVE TABU SEARCH ALGORITHM FOR
//                                   LOADING  CMS

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <conio.h>
#include <fstream.h>

// function definitions

void in_conf(int gmax_ic,int npart_ic,int nmach_ic,int x_ic[][15][8],
             int y_ic[][8],int z_ic[],int nop[]); /* function for reading the virtual cell
             configuration of the manufacturing shop */

void performa(int npart_p,int nmach_p,int gmax_p,int x_p[][15][8],int y_p[][8],
             int z_p[],int nop_p[],float per_vec[]); /* function to interact with
             simulation module and transfer performance measures */

void objfunc(int npart_o,int gmax_o,int nop_o[],int x_o[][15][8],int *objval); /* function to
             determine inter-cell part type transfer level */

void output(int npart_o,int nmach_o,int gmax_o,int nop_o[],int pre_sq_o[][15],
             int xopt_o[][15][8],float best_obj[]); /* function to print out part assignment
             and schedule */

void p_move(long idum3,int npart_pm,int gmax_pm,int nop_pm[],int
             mc_re_pm[][50], int pre_sq_pm[][15],int z_pm[],int x_pm[][15][8],int
             x1_pm[][15][8],int x2_pm[][15][8],int x3_pm[][15][8],int *i1,
             int *i2,int *i3); /* function to generate neighbourhood solutions */

void SIMANout(int npart_o2,int nmach_o2,int gmax_o2,int nop_o2[],
             int x_o2[][15][8], int y_o2[][8],int z_o2[]); /* function (translator) to
             generate and/or modify SIMAN experimental file for each neighbourhood
             solution */

void fpv(int npart_f,int nop_f[],int mc_f[][50], int pre_sq_f[][15],
```

```
                int gmax_f,int z_f[],int x_f[][15][8],int *constr1,int *constr2): /* function
                to check feasibility of solution vectors */

void fplv(long idum8,int npart_a,int nop_a[],int mc_a[][50],int pre_sq_a[][15],
                int gmax_a,int z_a[],int x_a[][15][8]); /* If the program used as a stand
                alone program for loading and scheduling this function generates initial
                feasible loading vector and/or can obtain initial solution from another data
                file */

void f1(); /* funtion to interface with simulation */

int FindValues(float* tardiness, float* utilization, float* total, float* mach1U,
                float* mach2U, float* mach3U, float* mach4U, float* mach5U,
                float* mach6U, float* mach7U, float* mach8U, float* mach9U,
                 float* mach10U, float* mach11U, float* mach12U); /* function to import
                simulation output */

float ran1(long *idum); /* function to generate random variables */
```

Frame D.2

```
//
//          MULTIPLE OBJECTIVE TABU SEARCH ALGORITHM
//                  FOR RECONFIGURING CMS
//

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <conio.h>
#include <fstream.h>

// function definitions

void feasible(int mmin_mf,int mmax_mf,int nmach_mf,int gmax_mf,int y1_mf[][8],
        int z1_mf[],int *gxytop, int *gtotal1, int *gtotal2); /* fuction to generate initial
        feasuble virtual cell configuration if we start from a random initial solution and/or
        we import the solution from MOCACEF 1.0*/

void in_conf(int gmax_ic,int npart_ic,int nmach_ic,int x_ic[][15][8],
        int y_ic[][8],int z_ic[],int nop[]); /* function to read initial virtual cell configuration
        */

void performa(int npart_p,int nmach_p,int gmax_p,int x_p[][15][8],int y_p[][8],
        int z_p[],int nop_p[],float per_vec[]); /* function to interact with
        simulation module and transfer performance measures */

void new_conf(float goal1,float goal2, float goal3, float goal4,
        long idum9,int mmin_mv,int mmax_mv,int nmach_mv,int npart_mv,
        int gmax_mv,int gmin_mv,int xopt_mv[][15][8],int yopt_mv[][8],
```

```
        int zopt_mv[],int y2_mv[][8],int z2_mv[],int x2_mv[][15][8],
        int y3_mv[][8],int z3_mv[],int x3_mv[][15][8],int y4_mv[][8],
        int z4_mv[],int x4_mv[][15][8],int nop_mv[],
        int tabusize,int tabulist[][300],
        float obj_vec2[],float obj_vec3[],float obj_vec4[]); /* funtion to generate
        neighbourhood solutions*/

void c_load(float goal1,int gmax_s,int npart_s,int nmach_s,int nop_s[],
        int x_s[][15][8],int y_s[][8],int z_s[],int xopt_s[][15][8],
        float bestve[]); /* function used by funtion new_conf while generating
        neighbourhood solution. This function assigns parts */

void objfunc(int npart_o,int gmax_o,int nop_o[],int x_o[][15][8],int *objval);

void output(int npart_o,int nmach_o,int gmax_o,int nop_o[],int xopt_o[][15][8],int
        yopt_o[][8],int zopt_o[],float best_obj[]);

void SIMANout(int npart_o2,int nmach_o2,int gmax_o2,int nop_o2[],int x_o2[][15][8],
        int y_o2[][8],int z_o2[]);

int FindValues(float* tardiness, float* utilization, float* total, float* mach1U,
        float* mach2U, float* mach3U, float* mach4U, float* mach5U, float* mach6U,
        float* mach7U, float* mach8U, float* mach9U, float* mach10U, float* mach11U,
        float* mach12U);

void f1();

float ran1(long *idum);
```

Appendix VII : Continuation of the experimental work form Chapter 9

In chapter 9, a good performing virtual cell configuration was found. Here, one more step is going to be taken. It is assumed that in the coming production run, there are some changes in the part list (i.e. a new part list). There are design changes on part types 4, 9 and 16 (see Table 7.4 in Chapter 7). Consequently, some of the processing requirements of these part types are changed and there is a demand for two new part types, namely part type 21 and 22. The new part list with the corresponding generic process plans is shown in Table VII.1 below.

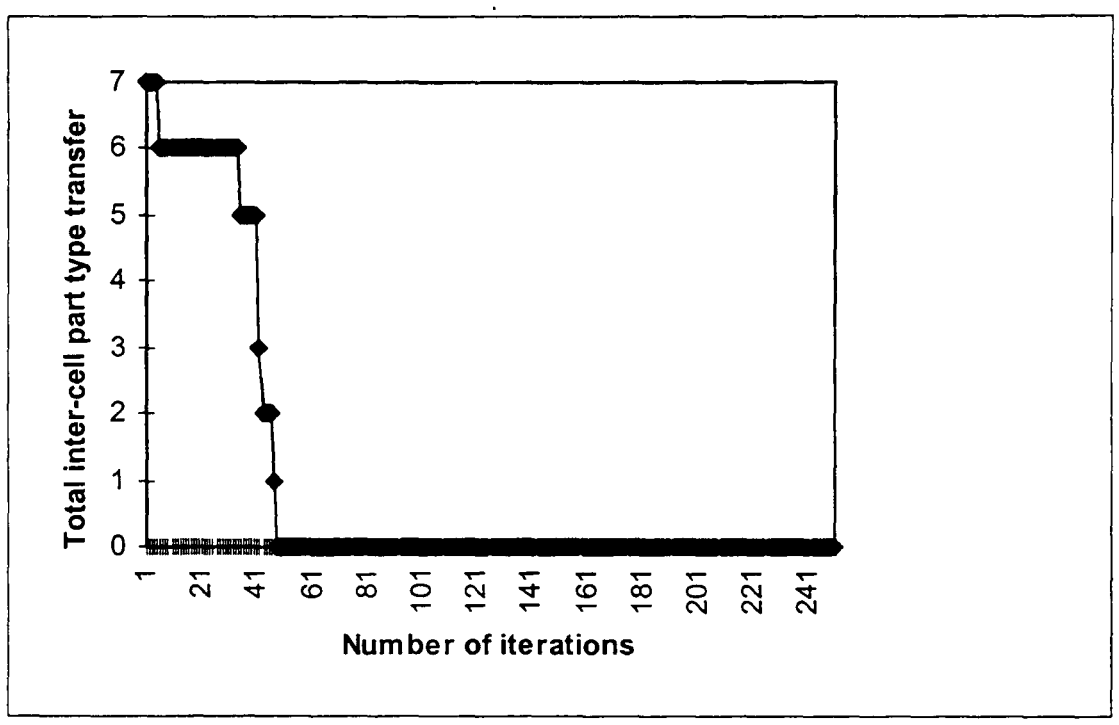
Table VII.1 The new part list and the correspond generic process plans

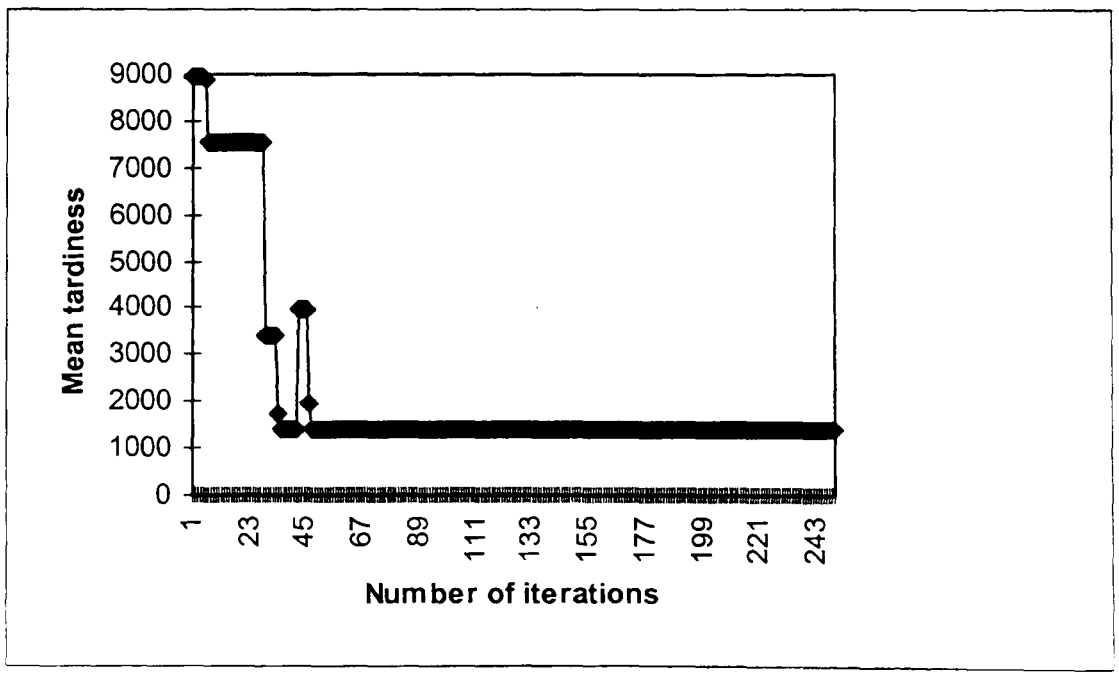
Part Type #	# Operations (NOP_i)	# RE	Operation # 1	Operation # 2	Operation # 3	Operation # 4
1	3	3	RE1(60)*	RE2(80)	RE4(90)	
2	3	3	RE1(50)	RE2(60)	RE3(40)	
3	3	3	RE5(20)	RE6(60)	RE7(80)	
4 (redesign)	3	3	RE8(40)	RE5(**40**)	**RE9(30)**	
5	3	3	RE7(50)	RE4(60)	RE5(80)	
6	3	3	RE8(50)	RE6(50)	RE7(60)	
7	3	3	RE8(70)	RE9(70)	RE10(80)	
8	3	3	RE9(60)	RE10(50)	RE11(90)	
9 (redesign)	3	3	RE5(70)	RE1(**30**)	**RE4(20)**	
10	2	2	RE3(40)	RE4(50)		
11	3	3	RE5(50)	RE6(50)	RE9(40)	
12	3	3	RE10(70)	RE8(80)	RE9(80)	
13	3	3	RE5(10)	RE8(30)	RE10(50)	
14	3	3	RE8(50)	RE7(40)	RE5(50)	
15	2	2	RE1(60)	RE2(20)		
16 (redesign)	3	3	RE3(**40**)	RE4(30)	**RE6(20)**	
17	3	3	RE6(40)	RE7(50)	RE8(70)	
18	4	4	RE8(50)	RE9(50)	RE10(50)	RE11(20)
19	2	2	RE5(50)	RE2(50)		
20	3	3	RE7(10)	RE8(50)	RE9(30)	
21(new)	4	4	**RE3(20)**	**RE4(25)**	**RE6(40)**	**RE10(50)**
22(new)	2	2	**RE5(50)**	**RE8(50)**		

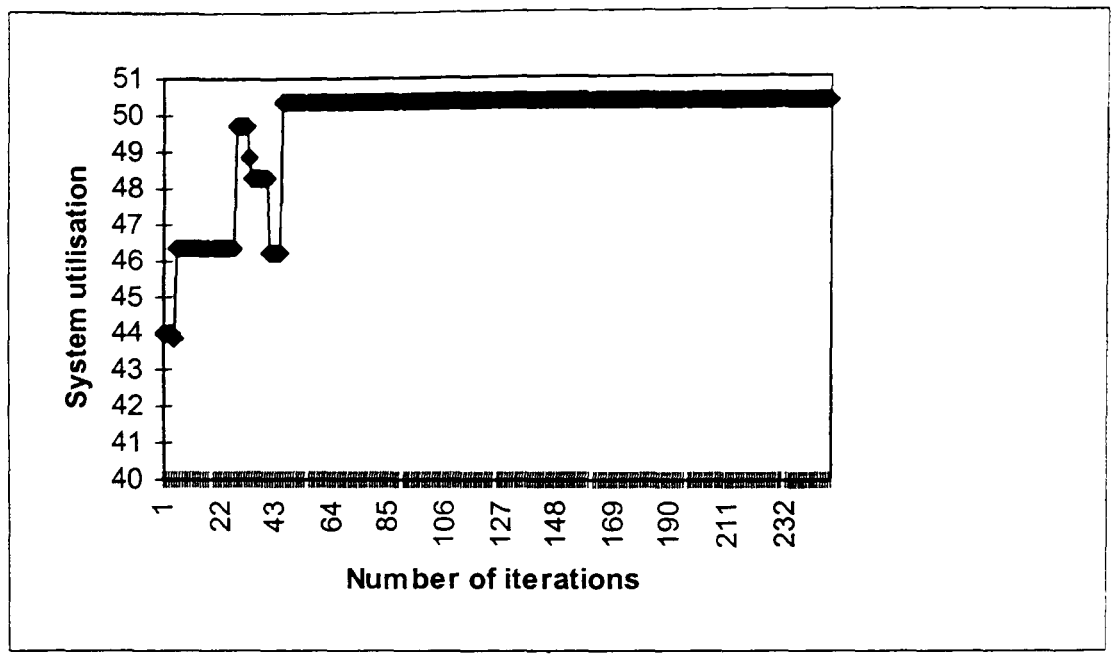
* RE#(A): Resource Element number (Processing time + set up time)

The loading module of the multiple objective decision support framework is first used to load the system. If the solution is satisfactory then it is going to be suggested for the implementation, if not a new set of virtual cells are going to be generated by

the reconfiguration module which may better satisfy the current demand. The decision-maker's goals were set to zero for inter-cell part type transfer, zero for mean tardiness, 65 percent for system utilisation and 5000 for total throughput. The loading module was iterated 250 times with 3 neighbour solutions in each iteration. The loading module converged to a solution in 125 minutes. The convergence graphics and the best solution found are shown in Figures VII.1 and VII.2 below.







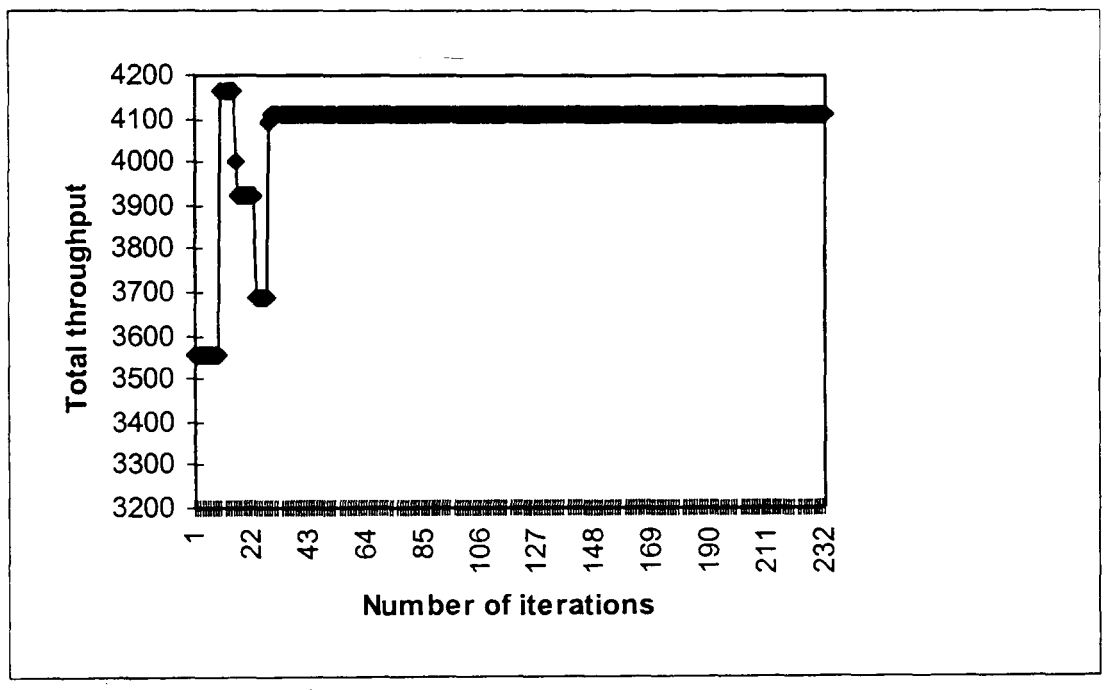


Figure VII.1 Conversion behaviour of the loading module

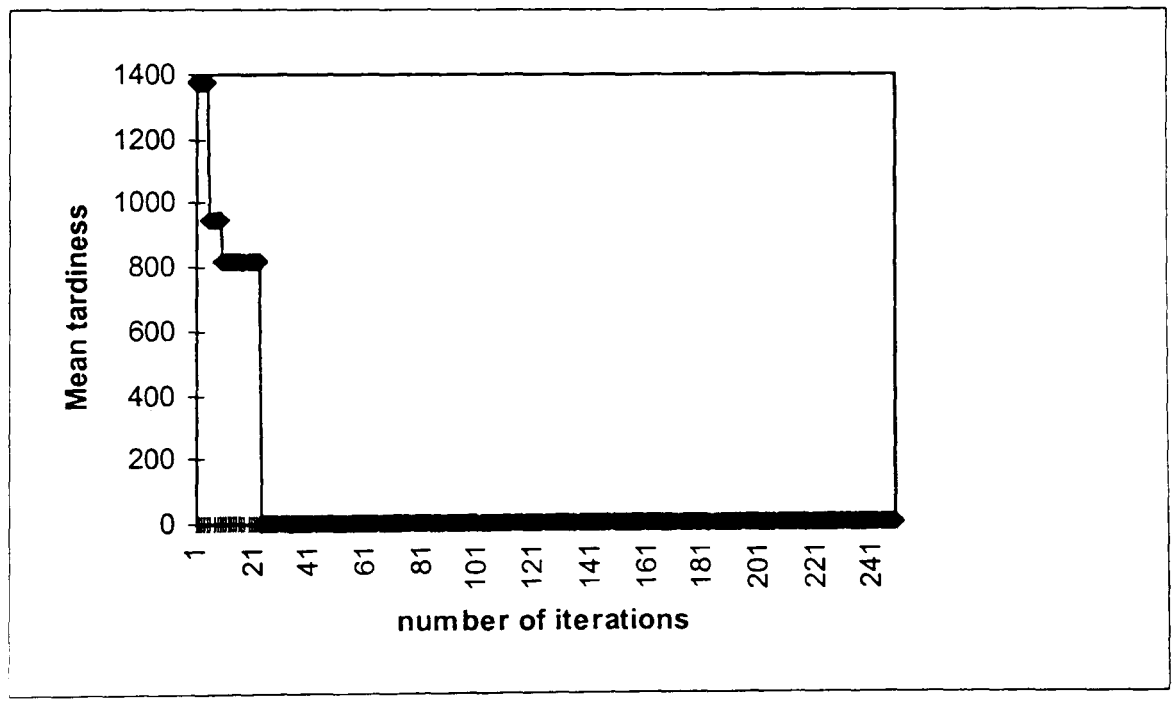
opt_load4a - Notepad

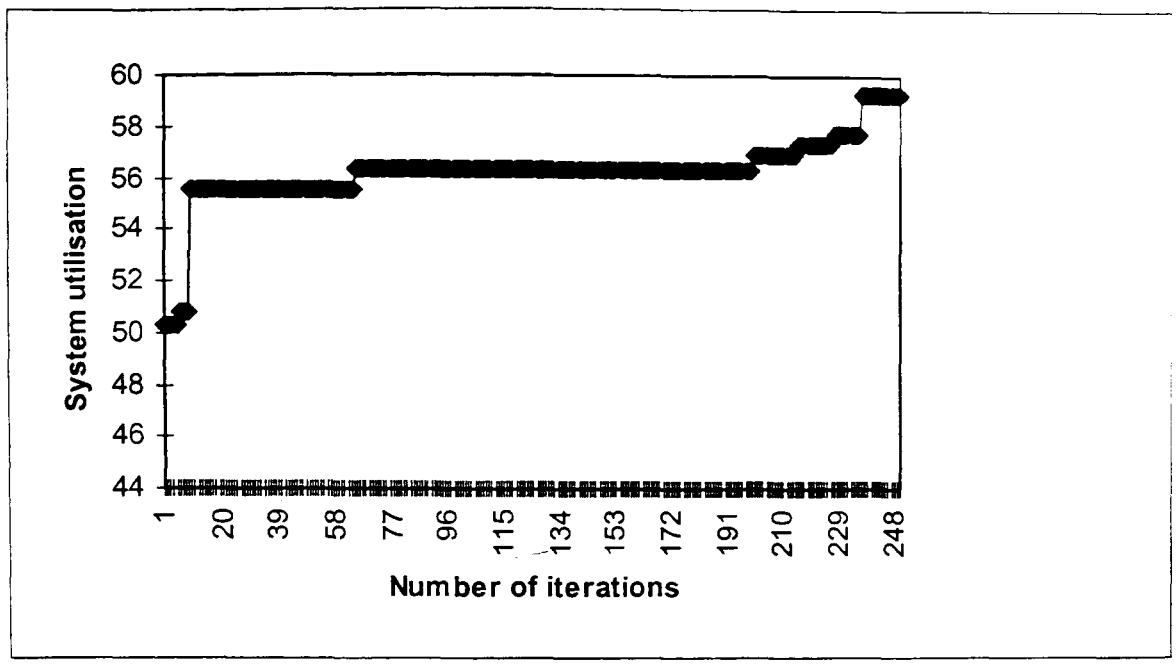
File Edit Search Help

```
==================================================
PROGRAM RE-MOCL

Multiple Objective Tabu Search Based Integrated
System for Loading CMS

By: Adil BAYKASOGLU
==================================================

Optimum Part Loading
--------------------
 Part Type->1 { 1 (3), 2 (3), 4 (3)}
 Part Type->2 { 1 (1), 2 (1), 3 (1)}
 Part Type->3 { 5 (1), 6 (1), 7 (1)}
 Part Type->4 { 8 (1), 5 (1), 9 (1)}
 Part Type->5 { 7 (1), 4 (1), 5 (1)}
 Part Type->6 { 8 (1), 6 (1), 7 (1)}
 Part Type->7 { 8 (4), 9 (4), 10 (4)}
 Part Type->8 { 9 (4), 10 (4), 11 (4)}
 Part Type->9 { 5 (1), 1 (1), 4 (1)}
 Part Type->10 { 3 (1), 4 (1)}
 Part Type->11 { 5 (1), 6 (1), 9 (1)}
 Part Type->12 { 10 (1), 8 (1), 9 (1)}
 Part Type->13 { 5 (4), 8 (4), 10 (4)}
 Part Type->14 { 8 (1), 7 (1), 5 (1)}
 Part Type->15 { 1 (3), 2 (3)}
 Part Type->16 { 3 (1), 4 (1), 6 (1)}
 Part Type->17 { 6 (1), 7 (1), 8 (1)}
 Part Type->18 { 8 (1), 9 (1), 10 (1), 11 (1)}
 Part Type->19 { 5 (2), 2 (2)}
 Part Type->20 { 7 (4), 8 (4), 9 (4)}
 Part Type->21 { 3 (1), 4 (1), 6 (1), 10 (1)}
 Part Type->22 { 5 (4), 8 (4)}


Some Performance Indicators

Total Intercell Traffic          =0.0
Mean Tardiness in the System     =1373.7
Overall System Utilisation       =50.33
Total Troughput from the System =4107
```

Figure VII.2 Part assignment scenario for the best solution found

As the results showed, only the inter-cell movement goal was met with the existing virtual cell configuration. Other objectives are not satisfied. Therefore, the reconfiguration module was executed to generate a new virtual cell configuration that could better satisfy the decision-maker's goals. The reconfiguration module was iterated 250 times with 3 neighbour solutions in each iteration. The reconfiguration module converged to a better solution in 124 minutes. The convergence graphics, the best solution obtained and a part of the generated optimal schedule are shown in Figures VII.3, VII.4 and VII.5 below. The input to the reconfiguration module are the best solution obtained from the loading module and the current virtual cell configuration. Therefore, reconfiguration module tries to improve this solution. Inter-cell movement goal was satisfied in the loading module and, it was considered as the most important objective therefore it is not going to get worse in the reconfiguration module, but the other objectives will improve if possible.





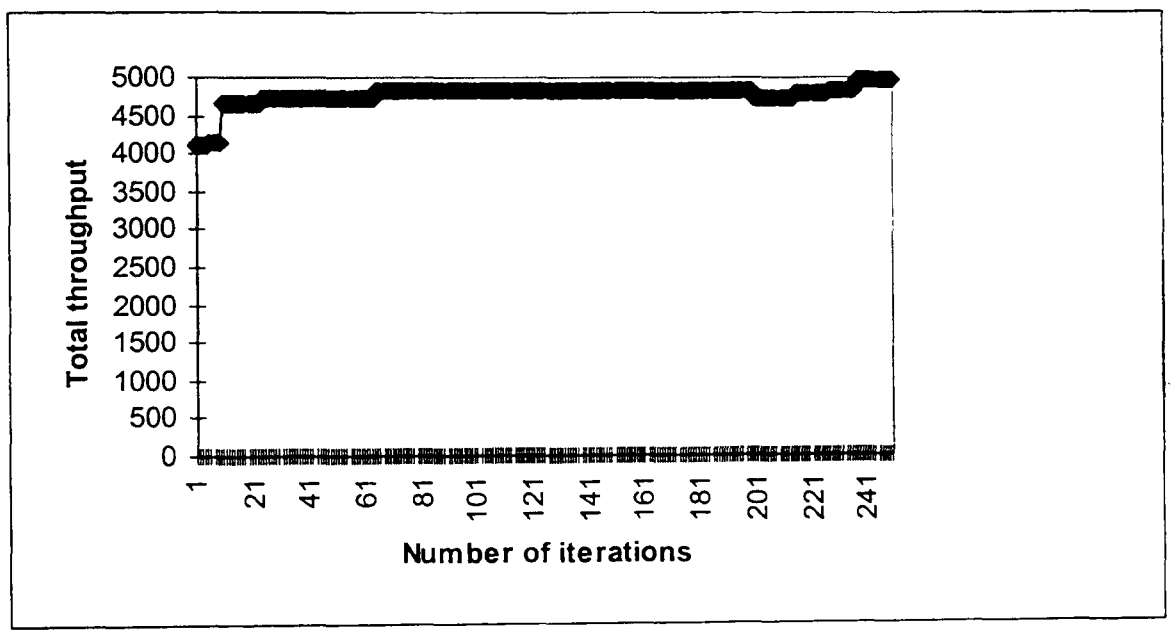


Figure VII.3 Conversion behaviour of the reconfiguration module

OUTPUT OF THE RECONFIGURATION PROGRAM
(NEW VCs & PART ASSIGNMENT)

VCs after Reconfiguration

Virtual Cell->1
Machine->3
Machine->6
Machine->8
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->3, RE->4, RE->5, RE->6, RE->7, RE->8, RE->9, RE->10, RE->11
Virtual Cell->2
Machine->2
Machine->4
Machine->9
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->3, RE->4, RE->5, RE->6, RE->7, RE->8, RE->9, RE->10, RE->11
Virtual Cell->3
Machine->1
Machine->5
Machine->7
Machine->10
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->3, RE->4, RE->6, RE->7, RE->8, RE->9, RE->10, RE->11
Virtual Cell->4
Machine->11
Machine->12
Resource Elements in the Virtual Cell
RE->1, RE->2, RE->4, RE->5, RE->7

Part Loading (assignment) for new VCs

Part Type->1 { 1 (4), 2 (4), 4 (4)}
Part Type->2 { 1 (1), 2 (1), 3 (1)}
Part Type->3 { 5 (1), 6 (1), 7 (1)}
Part Type->4 { 8 (2), 5 (2), 9 (2)}
Part Type->5 { 7 (2), 4 (2), 5 (2)}
Part Type->6 { 8 (3), 6 (3), 7 (3)}
Part Type->7 { 8 (2), 9 (2), 10 (2)}
Part Type->8 { 9 (1), 10 (1), 11 (1)}
Part Type->9 { 5 (4), 1 (4), 4 (4)}
Part Type->10 { 3 (2), 4 (2)}
Part Type->11 { 5 (1), 6 (1), 9 (1)}
Part Type->12 { 10 (3), 8 (3), 9 (3)}
Part Type->13 { 5 (2), 8 (2), 10 (2)}
Part Type->14 { 8 (1), 7 (1), 5 (1)}
Part Type->15 { 1 (1), 2 (1)}
Part Type->16 { 3 (3), 4 (3), 6 (3)}
Part Type->17 { 6 (2), 7 (2), 8 (2)}
Part Type->18 { 8 (3), 9 (3), 10 (3), 11 (3)}
Part Type->19 { 5 (2), 2 (2)}
Part Type->20 { 7 (1), 8 (1), 9 (1)}
Part Type->21 { 3 (3), 4 (3), 6 (3), 10 (3)}
Part Type->22 { 5 (1), 8 (1)}

Figure VII.4 The output of the reconfiguration module: new VCs and corresponding part assignments

Sim_out - WordPad

File Edit View Insert Format Help

Courier New (Western) 10

```
PartID/Type  2/13 OperationNo:  1 V_Cell: 2 Machine:  9 RE: 5 Start:  0 Finish: 10
PartID/Type  2/13 OperationNo:  2 V_Cell: 2 Machine:  2 RE: 8 Start: 10 Finish: 40
PartID/Type  4/ 5 OperationNo:  1 V_Cell: 2 Machine:  4 RE: 7 Start: 28 Finish: 78
PartID/Type  6/19 OperationNo:  1 V_Cell: 2 Machine:  9 RE: 5 Start: 37 Finish: 87
PartID/Type  2/13 OperationNo:  3 V_Cell: 2 Machine:  2 RE: 10 Start: 41 Finish: 91
PartID/Type  8/20 OperationNo:  1 V_Cell: 1 Machine:  3 RE: 7 Start: 48 Finish: 58
PartID/Type  8/20 OperationNo:  2 V_Cell: 1 Machine:  8 RE: 8 Start: 58 Finish: 108
PartID/Type 10/ 6 OperationNo:  1 V_Cell: 3 Machine:  5 RE: 8 Start: 58 Finish: 108
PartID/Type 12/ 5 OperationNo:  1 V_Cell: 2 Machine:  4 RE: 7 Start: 78 Finish: 128
PartID/Type  6/19 OperationNo:  2 V_Cell: 2 Machine:  2 RE: 2 Start: 91 Finish: 141
PartID/Type 14/18 OperationNo:  1 V_Cell: 3 Machine:  5 RE: 8 Start: 108 Finish: 158
PartID/Type  8/20 OperationNo:  3 V_Cell: 1 Machine:  8 RE: 9 Start: 108 Finish: 138
PartID/Type 10/ 6 OperationNo:  2 V_Cell: 3 Machine: 10 RE: 6 Start: 109 Finish: 159
PartID/Type 18/11 OperationNo:  1 V_Cell: 1 Machine:  6 RE: 5 Start: 124 Finish: 174
PartID/Type  4/ 5 OperationNo:  2 V_Cell: 2 Machine:  4 RE: 4 Start: 128 Finish: 188
PartID/Type 22/ 4 OperationNo:  1 V_Cell: 2 Machine:  2 RE: 8 Start: 144 Finish: 184
PartID/Type 24/17 OperationNo:  1 V_Cell: 2 Machine:  9 RE: 6 Start: 146 Finish: 186
PartID/Type 16/18 OperationNo:  1 V_Cell: 3 Machine:  5 RE: 8 Start: 158 Finish: 208
PartID/Type 10/ 6 OperationNo:  3 V_Cell: 3 Machine:  7 RE: 7 Start: 159 Finish: 219
PartID/Type 26/14 OperationNo:  1 V_Cell: 1 Machine:  8 RE: 8 Start: 167 Finish: 217
PartID/Type 18/11 OperationNo:  2 V_Cell: 1 Machine:  6 RE: 6 Start: 174 Finish: 224
PartID/Type 22/ 4 OperationNo:  2 V_Cell: 2 Machine:  9 RE: 5 Start: 186 Finish: 226
PartID/Type 24/17 OperationNo:  2 V_Cell: 2 Machine:  2 RE: 7 Start: 187 Finish: 237
PartID/Type 12/ 5 OperationNo:  2 V_Cell: 2 Machine:  4 RE: 4 Start: 188 Finish: 248
PartID/Type 20/16 OperationNo:  1 V_Cell: 3 Machine:  5 RE: 3 Start: 208 Finish: 248
PartID/Type 26/14 OperationNo:  2 V_Cell: 1 Machine:  8 RE: 7 Start: 217 Finish: 257
PartID/Type  4/ 5 OperationNo:  3 V_Cell: 2 Machine:  9 RE: 5 Start: 226 Finish: 306
PartID/Type 22/ 4 OperationNo:  3 V_Cell: 2 Machine:  2 RE: 9 Start: 237 Finish: 267
PartID/Type 14/18 OperationNo:  2 V_Cell: 3 Machine:  5 RE: 9 Start: 248 Finish: 298
PartID/Type 20/16 OperationNo:  2 V_Cell: 3 Machine:  7 RE: 4 Start: 249 Finish: 279
PartID/Type 18/11 OperationNo:  3 V_Cell: 1 Machine:  8 RE: 9 Start: 257 Finish: 297
PartID/Type 26/14 OperationNo:  3 V_Cell: 1 Machine:  6 RE: 5 Start: 257 Finish: 307
PartID/Type 24/17 OperationNo:  3 V_Cell: 2 Machine:  2 RE: 8 Start: 267 Finish: 337
PartID/Type 20/16 OperationNo:  3 V_Cell: 3 Machine: 10 RE: 6 Start: 279 Finish: 299
PartID/Type 16/18 OperationNo:  2 V_Cell: 3 Machine:  5 RE: 9 Start: 298 Finish: 348
PartID/Type 14/18 OperationNo:  3 V_Cell: 3 Machine:  7 RE: 10 Start: 299 Finish: 349
PartID/Type 12/ 5 OperationNo:  3 V_Cell: 2 Machine:  9 RE: 5 Start: 306 Finish: 386
PartID/Type 32/ 3 OperationNo:  1 V_Cell: 1 Machine:  6 RE: 5 Start: 307 Finish: 327
```

NUM

For Help, press F1

Figure VII.5 A portion of the production schedule generated automatically by the reconfiguration module

The interpretation of the results show that inter-cell movement and tardiness goals have been satisfied as the most prioritised objectives. The system utilisation level is improved from 50.33% to 60% in the new VC configuration, which is now closer to the decision-maker's target. The total throughput has been improved from 4107 to 4973 which is also very close to the decision-maker's target.